University of Economics in Prague

Faculty of Informatics and Statistics

Department of Information and Knowledge Engineering

Doctoral Dissertation Thesis

Unsupervised Entity Classification
with Wikipedia and WordNet

PhD Candidate : Ing. Tomáš Kliegr
Advisor : Prof. RNDr. Jan Rauch, CSc.
Field of Study : Computer science (Informatika)


```
@PHDTHESIS{Kliegr-2012-PhdThesis,
    AUTHOR = {Tom{\'a}\v{s} Kliegr},
    TITLE = {{Unsupervised Entity Classification
              with Wikipedia and WordNet}},
    SCHOOL = {{V\v{S}E v Praze}},
    YEAR = {2012},
    TYPE = {Dissertation}
}
```

Prague, July 2012

Printed on recycled paper Written & coded with free software Typed with TEX

Prohlášení

Prohlašuji, že doktorskou práci na téma "Unsupervised Entity Classification with Wikipedia and WordNet" jsem vypracoval samostatně. Použitou literaturu a podkladové materiály uvádím v přiloženém seznamu literatury.

V Praze dne 31.7.2012

. . . . . .

Tomáš Kliegr

to Krishna

Acknowledgments

I would like to thank my supervisor, Prof. Jan Rauch, for his patience, advise and encouragement, without which this dissertation would never be finished. The seed idea for entity classification using Wikipedia and WordNet came from a paper by my colleague and friend Jan Nemrava just before I started scholarship at the Queen Mary University, London. It was there, where the first tangible output came to light. Krishna Chandramouli played an indispensable role in motivating the research by practical use case in image retrieval. Dario, Krishna and many other colleagues set an example to follow by their hard work and the number of lab hours, day or night, devoted to tackling research problems.

Essential part of the dissertation was written and coded after my return to Prague. I am thankful to Vojtěch Svátek for engaging in frequent discussions and periodically polling for a progress update. Following the example set by Ondřej Zamazal, this dissertation was written and developed using free software.

My research was supported by the following projects: two EU Networks of Excellence: K-Space and PetaMedia, and the ongoing EU integrated project LinkedTV.

Finally, I would like to thank my wife Lucia and František Květoslav for their love and support.

Abstrakt

Dizertační práce se věnuje problému klasifikace entit reprezentovaných jmennými frázemi v textu. Cílem je vyvinout metodu pro automatizovanou klasifikaci těchto entit v datasetech skládajících se z krátkých textových fragmentů. Důraz je kladen na metody učení bez učitele, nebo kombinaci učení s učitelem a bez učitele (angl. semi-supervised learning), přičemž nebudou vyžadovány trénovací příklady. Třídy jsou buď automaticky stanoveny nebo zadány uživatelem.

Náš první pokus pro řešení problému klasifikace entit je algoritmus Sémantického Mapování Konceptů (angl. Semantic Concept Mapping – SCM). Tento algoritmus mapuje jmenné fráze i cílové třídy na koncepty thesauru WordNet. Grafové míry podobnosti pro WordNet jsou použity pro přiřazení nejbližší třídy k dané jmenné frázi. Pokud jmenná fráze není namapována na žádný koncept, potom je použit algoritmus Cíleného Objevování Hyperonym (angl. Targeted Hypernym Discovery – THD). Tento algoritmus extrahuje s pomocí lexiko-syntaktických vzorů hyperonymum z článku na Wikipedii, který danou jmennou frázi definuje. Toto hyperonymum je použito k namapování jmenné fráze na koncept ve WordNetu. Hyperonymum může být samo o sobě také považováno za výsledek klasifikace. V takovém případě je dosaženo klasifikace bez učitele.

Algoritmy SCM a THD byly navrženy pro angličtinu. I když je možné oba algoritmy přizpůsobit i pro jiné jazyky, byl v rámci dizertační práce vyvinut algoritmus *Pytel článků* (angl. Bag of Articles – BOA), který je jazykově agnostický, protože je založen na statistickém Rocchio klasifikátoru. Díky zapojení Wikipedie jako zdroje informací pro klasifikaci nevyžaduje BOA trénovací data. WordNet je využit novým způsobem, a to pro výpočet vah slov, jako pozitivní seznam slov a pro lematizaci. Byl také navržen disambiguační algoritmus pracující s globálním kontextem. Algoritmus BOA považujeme za hlavní přínos dizertace.

Experimentální hodnocení navržených algoritmů je provedeno na datasetu WordSim353 používaném pro hodnocení systémů pro výpočet podobnosti slov (angl. Word Similarity Computation – WSC), a na datasetu Český cestovatel, který byl vytvořen speciálně pro účel našeho výzkumu. Na datasetu WordSim353 dosahuje BOA Spearmanova korelačního koeficientu 0.72 s lidským hodnocením. Tento výsledek je blízko hodnotě 0.75 dosažené algoritmem ESA, který je podle znalosti autora nejlepším algoritmem pro daný dataset nevyžadujícím trénovací data. Algoritmus BOA je ale výrazně méně náročný na předzpracování Wikipedie než ESA.

Algoritmus SCM nedosahuje dobrých výsledků na datasetu WordSim353, ale naopak předčí BOA na datasetu Český cestovatel, který byl navržen speciálně pro úlohu klasifikace entit. Tato nesrovnalost vyžaduje další výzkum. V samostatném hodnocení THD na malém počtu pojmenovaných entit z datasetu Český cestovatel bylo správné hyperonymum nalezeno v 62% případů.

Další dosažené výsledky samostatného významu zahrnují novou funkci pro vážení slov založenou na WordNetu, kvalitativní a kvantitativní vyhodnocení možností využití Wikipedie jako zdroje textů pro objevování hyperonym s využitím lexiko-syntaktických vzorů a zevrubnou rešerši měr podobnosti nad WordNetem zahrnující též jejich výkonnostní porovnání na datasetech WordSim353 a Český cestovatel.

Abstract

This dissertation addresses the problem of classification of entities in text represented by noun phrases. The goal of this thesis is to develop a method for automated classification of entities appearing in datasets consisting of short textual fragments. The emphasis is on unsupervised and semi-supervised methods that will allow for fine-grained character of the assigned classes and require no labeled instances for training. The set of target classes is either user-defined or determined automatically.

Our initial attempt to address the entity classification problem is called *Semantic Concept Mapping* (SCM) algorithm. SCM maps the noun phrases representing the entities as well as the target classes to WordNet. Graph-based WordNet similarity measures are used to assign the closest class to the noun phrase. If a noun phrase does not match any WordNet concept, a *Targeted Hypernym Discovery* (THD) algorithm is executed. The THD algorithm extracts a hypernym from a Wikipedia article defining the noun phrase using lexico-syntactic patterns. This hypernym is then used to map the noun phrase to a WordNet synset, but it can also be perceived as the classification result by itself, resulting in an unsupervised classification system.

SCM and THD algorithms were designed for English. While adaptation of these algorithms for other languages is conceivable, we decided to develop the *Bag of Articles* (BOA) algorithm, which is language agnostic as it is based on the statistical Rocchio classifier. Since this algorithm utilizes Wikipedia as a source of data for classification, it does not require any labeled training instances. WordNet is used in a novel way to compute term weights. It is also used as a positive term list and for lemmatization. A disambiguation algorithm utilizing global context is also proposed. We consider the BOA algorithm to be the main contribution of this dissertation.

Experimental evaluation of the proposed algorithms is performed on the WordSim353 dataset, which is used for evaluation in the Word Similarity Computation (WSC) task, and on the Czech Traveler dataset, the latter being specifically designed for the purpose of our research. BOA performance on WordSim353 achieves Spearman correlation of 0.72 with human judgment, which is close to the 0.75 correlation for the ESA algorithm, to the author's knowledge the best performing algorithm for this gold-standard dataset, which does not require training data. The advantage of BOA over ESA is that it has smaller requirements on preprocessing of the Wikipedia data.

While SCM underperforms on the WordSim353 dataset, it overtakes BOA on the Czech Traveler dataset, which was designed specifically for our entity classification problem. This discrepancy requires further investigation. In a standalone evaluation of THD on Czech Traveler dataset the algorithm returned a correct hypernym for 62% of entities.

The additional results of possibly separate interest include a novel WordNet-based term-weighting function, qualitative and quantitative evaluation of Wikipedia as a textual resource for discovering hypernyms with Hearst patterns, and a comprehensive survey of WordNet similarity measures including their benchmark on both WordSim353 and Czech Traveler datasets.

Resumen

El objetivo de esta tesis es desarrollar un método para la clasificación automática de las entidades que aparecen en conjuntos de datos basados en fragmentos cortos de texto. El énfasis se situará en los métodos sin supervisión y semi-supervisados, que permitirán una granularidad fina de caracteres de las clases de destino, y no requieren instancias etiquetadas para el aprendizaje.

Nuestro intento inicial de abordar el problema de la clasificación de entidades se denomina algoritmo de Mapeo Semántico de Conceptos (SCM por sus siglas en inglés). El algoritmo SCM mapea las oraciones sustantivas que representan las entidades y las clases de destino a WordNet. Entonces se emplea una medida de similaridad de WordNet para asignar la clase más próxima a la oración sustantiva. Si no se puede establecer una correspondencia entre una oración sustantiva con algún concepto de WordNet, se ejecutará un algoritmo de Descubrimiento Dirigido de Hiperónimo (THD por sus siglas en inglés). El algoritmo THD extrae un hiperónimo de un artículo de la Wikipedia definiendo la oración sustantiva usando patrones léxico-sintácticos. Este hiperónimo se usa para mapear la oración sustantiva a un grupo de sinónimos cognitivos de WordNet. El hiperónimo puede ser percibido también como el resultado de la clasificación en si mismo, obteniendo así un sistema de clasificación sin supervisión.

Los algoritmos SCM y THD fueron diseñados para el inglés. Mientras que la adaptación de estos algoritmos a otros idiomas es concebible, hemos decidido desarrollar el algoritmo Saco de Artículos (BOA por sus siglas en inglés), que es agnóstico con respecto al idioma y se basa en el clasificador estadístico de Rocchio. Este algoritmo también usa la Wikipedia para la clasificación. WordNet se usa de un modo nuevo para calcular los pesos de los términos. También se emplea como una lista de términos positivos y para lematizar. Se propone asimismo un algoritmo de desambiguación opcional que emplea un contexto global. Consideramos el algoritmo BOA como la principal contribución de esta disertación.

La evaluación experimental de estos algoritmos se ha llevado a cabo sobre el conjunto de datos WordSim353 y sobre el conjunto Czech Traveler, este último específicamente diseñado para el propósito de nuestra investigación. BOA obtiene una correlación Spearman de 0.72 con supervisión humana, que es próxima a la correlación 0.75 para el algoritmo ESA, que hasta donde el autor sabe se trata del único y mejor algoritmo en rendimiento para este conjunto de datos de referencia.

Mientras que SCM rinde por debajo sobre WordSim353, sobrepasa a BOA en el conjunto de datos Czech Traveler. Esta discrepancia requiere más investigación. En una evaluación aislada de THD sobre el conjunto de datos Czech Traveler el algoritmo devolvió el hiperónimo correcto para el 62% de las entidades.

Los resultados adicionales de posible interés por separado incluyen una nueva función de ponderado de términos basada en WordNet, la evaluación cualitativa y cuantitativa de la Wikipedia como una fuente textual para el descubrimiento de hiperónimos con patrones de Hearst, y un estudio exhaustivo de las medidas de similaridad de WordNet incluyendo su banco de pruebas sobre los conjuntos de datos WordSim353 y Czech Traveler.

Contents

List of Figures

List of Tables

List of Algorithms

List of Examples

Introduction

This dissertation addresses the problem of extraction and classification of entities represented by noun phrases. This task, which we call *entity classification*, arises for example in the analysis of image captions. While techniques for Named Entity Recognition and classification (NER) are well-researched, NER classifiers typically need to be trained on large labeled document corpora, which generally involve only several labels. The goal of this thesis is to develop a method for automated classification of entities appearing in datasets consisting of short textual fragments. The emphasis is on unsupervised and semi-supervised methods that will allow for fine-grained character of target classes.

Since we consider entities to be nouns or noun phrases coming from short textual fragments, we cannot assume to have their left and right *local* context available for n-gram techniques to be applied. On the other hand, it is often the case that unlabeled entities coming from the same dataset share the same *global* context. Such properties are often exhibited by captions of images that come from the same collection. This raises a need for a disambiguation algorithm, which can take advantage of the global context.

The goal of this dissertation is to develop a solution for entity classification that would:

- holistically address the entity classification problem:
  - extract entities from plain text,
  - accept user-defined set of target classes or no set of classes at all,
  - require no training set from the user,
- have results comparable with the state-of-the-art algorithms,
- disambiguate using global context,
- follow zeitgeist maintaining near real time performance – new named entities are recognized once they are covered by Wikipedia, the total time to process one input entity and a set of ten or less target classes takes several seconds at most,
- use English as the target language, but allow for easy extensibility to other Indo-European languages.

Our initial attempt to address the entity classification problem, conceived already in 2008, is presented in Chapter 1. This algorithm, called *Semantic Concept Mapping* (SCM) approaches the entity classification task by mapping the noun phrases representing the entities as well as the target classes to WordNet synsets. A WordNet similarity measure is then used to assign the closest class to the noun phrase. If a noun phrase does not match any WordNet synset, a *Targeted Hypernym Discovery* (THD) algorithm is executed: a Wikipedia article defining the noun phrase is retrieved using a hybrid measure based on article popularity and text-based relevance. Hypernym from this article is extracted using a rule-based extraction grammar. This hypernym is then used to map the noun phrase to a WordNet synset. The hypernym can

be also perceived as the classification result by itself, resulting in an unsupervised classification system.

Experiments presented in our paper [KCN+08a] showed that Wikipedia fares remarkably well in the THD task, while WordNet-based classification did not produce satisfactory results. SCM algorithm also does not yet exploit the global context.

In response, we proposed an outline for the Bag-Of-Articles (BOA) algorithm in [Kli10] in 2010. This algorithm, presented in detail in Chapter 2, uses Wikipedia also for classification, and the use of WordNet is diminished to an optional term-weighting function, positive term list and lemmatization. An optional disambiguation algorithm utilizing global context is also proposed. We consider the BOA algorithm to be the main contribution of this dissertation.

The related research is placed after the description of the SCM and BOA algorithms into Chapter 3. It should be noted that the first papers on Wikipedia-based word similarity computation, which is the closest field to our focus, started to appear around 2007, when also this dissertation was started. Chapter 3 thus serves primarily for comparison of other work with our contribution, with SCM and BOA algorithms being at this point known to the reader from the previous chapters, rather than as a description of foundations on top of which our algorithms are built.

Both our algorithms – SCM and BOA – use WordNet and Wikipedia, albeit in different ways and to different extent. The discussion of these two knowledge sources is solicited into two separate chapters. Chapter 4 focuses on WordNet with emphasis on its use for word similarity computation. The similarity measures introduced in this chapter are central for the SCM algorithm. They are also relevant for term-weighting functions proposed for use in the BOA algorithm.

Chapter 5 examines the potential of using Wikipedia for THD and discusses the factors influencing the existence of a Wikipedia page covering an entity. An important part of this chapter are new experimental results that give a quantitative perspective of the issues raised.

Chapter 6 presents evaluation of the proposed algorithms. This chapter also gives a review of relevant experimental datasets and introduces the Czech Traveler dataset, which was specifically developed for the purpose of our research. Experimental evaluation is performed on the WordSim353 dataset, the most commonly referenced one in the field of word similarity computation, and on the Czech Traveler dataset.

The conclusions located in Chapter 7 summarize our contribution. The dissertation is finished with several appendices. The grammar used by our THD algorithm is presented in Appendix A, configuration of the software implementation is described in Appendix B, Appendix C reprints the standard benchmark in the WSC task – the WordSim353 dataset and Appendix D introduces the Czech Traveler dataset.

1. Semantic Concept Mapping

The goal of this chapter is to introduce our first attempt to address the entity classification problem, which we call the Semantic Concept Mapping (SCM) algorithm. SCM takes on its input a list of target classes, represented as concepts from a thesaurus, and a textual fragment. It breaks down the textual fragment to noun phrases, which are assumed to correspond to entities, and then maps these entities also to thesaurus entries. SCM performs classification by exploiting the relations between common entities codified in the thesaurus. For this purpose it relies on already existing and well-researched similarity measures and the WordNet thesaurus, no labeled data are involved. The winning class has the highest value of the selected similarity measure with the entity.

If an entity cannot be mapped to an entry in the thesaurus, the system tries to map it to a thesaurus entry using targeted hypernym discovery. THD builds upon body of available work on discovery of hypernyms with lexico-syntactic patterns from text. It is called *targeted*, because it does not extract all word-hypernym pairs like many other approaches, but its input is only the entity for which the hypernym should be discovered. The algorithm tries to identify the most suitable document describing the entity, and in this document the most likely hypernym for the entity. The documents defining the entity are searched in an encyclopedic resource and lexico-syntactic patterns are used to extract the hypernym.

For the sake of clarity and practicality, we make some decisions regarding the knowledge sources and technologies employed. WordNet is used as a thesaurus, Wikipedia as an encyclopedic resource and the General Framework for Text Engineering (GATE) [CMBT02] as the linguistic framework. Since these choices represent the de facto standard in some fields of applied NLP, and there is an abundance of literature on these resources, we believe that it is not necessary to describe them in detail.

This chapter is organized as follows. Sec. 1.1 gives the motivation for the SCM method. The technical workflow of the algorithm is covered in Sec. 1.2. If the entity cannot be mapped in a straightforward way, THD is executed as detailed in Sec. 1.3. Sec. 1.4 gives an example. The implementation is described in Sec. 1.5 and the experimental results are summarized in Sec. 1.6.

Also note that the SCM method relies on WordNet similarity measures, these are described in Chapter 4. Chapter 5 focuses on Wikipedia as a resource for THD.

1.1. Motivation

In the discussion of an algorithm for classification of textual entities we can draw some parallels with the image classification task. Image classification algorithms typically perform classification of whole images or image segments to multiple categories with the choice of categories varying from application to application. The classifier typically needs to be retrained each time the set of categories changes. This is not an unsurmountable problem since the amount of required training data is usually small enough to make the design of human-labeled training

sets feasible. The resulting classifiers often have soft outputs, assigning a confidence measure to the prediction of class label.

Example 1.1 (Human labels entities in image annotation). A human evaluator, Helen, is asked to classify objects appearing in image annotation "lunch in a park surrounding Livadia Palace" to the following classes: C = *sand, sea, vegetation, person, sky, rock, tree, grass, ground, building*. Helen identifies entities "lunch", "park" and "Livadia Palace". For "lunch", no concept seems similar enough, she can either decide no to mark it -or- to choose almost arbitrarily any concept. For "park", Helen arbitrarily decides between "vegetation" and "grass" and proceeds to the last noun phrase "Livadia Palace". Not seeing this exact sequence of two words before, she hesitates between looking the entity up in Wikipedia and judging about the entity only from the head noun. Opting for the quicker latter option, she unanimously selects "building" as the target class, since it is conceptually closest to "palace"; palace is a kind of building. [a]

[a] After the evaluation, Helen may complain about the unfitting set of classes. The classes picked for the example were the same as used for labeling *image segments* in [KCN+08a].

In contrast, images are typically represented with much lower number of features, e.g. [WMS00] note that typical descriptor dimensions range from few tens to several hundreds. In addition, the feature vectors of individual annotations are sparse; [Wes00] reports a ratio of 158:1 between the average number of terms in an image caption and the total number of terms in the collection of captions. Data sparsity has a particularly severe effect on uncommon words including named entities, which are often of central importance in image annotations.

With SCM we take a human inspired – rather than statistical – approach to entity classification as illustrated by Example 1.1.

1.2. Workflow

Our algorithm proceeds in a similar way as a human would if presented an image annotation, a set of target classes C, and asked to express what is probably on the image using only the concepts provided:

- The algorithm first identifies the likely objects on the image by parsing the annotation for entities with `extractNounphrases` method. We assume each *noun phrase* in the input text to be an entity. Noun phrase is understood simply as a linguistic unit that generally consists of a noun preceded by zero to multiple modifiers.

- Entities are mapped to entries in the WordNet thesaurus with the `wordnetMap` method. This method has internally several branches:
  - For noun phrases that directly match one entry in the WordNet thesaurus, the mapping is straightforward.
  - If there are multiple matching entries, *disambiguation* by selecting the most frequent sense is performed.
  - If no entry is matching even the *head noun* (the last noun in the noun phrase), the system uses targeted hypernym discovery from Wikipedia to replace the original noun phrase with its hypernym.

- The target classes are also represented as WordNet entries. Once the entities are mapped to WordNet, for each entity function `wordnetSim` selects the closest target class in terms of "distance" in the WordNet thesaurus.

Alg. 1 provides a high level view of SCM classification.

Algorithm 1 Semantic Concept Mapping

Input: textual fragment $TEXT$, set of concepts C
Output: Set T of tuples $\langle np, c\rangle, c \in C, np$ noun phrase occurring in $TEXT$

$T := \emptyset$
NP:= extractNounphrases($TEXT$)
for all noun phrase $np \in NP$ **do**
  $wordnetMap$:= mapToWordNet(np)
  $class$:= $\arg\max_{c \in C}\ wordnetSim(wordnetMap, c)$
  $T := T \cup \{\langle np, class\rangle\}$
end for
return T

In the following, the significant parts of this algorithm will be discussed in detail. The noun phrase extraction `extractNounphrases` is done by the Noun Phrase Chunker extension of the GATE NLP framework. The `mapToWordNet` method is covered in Subs. 1.2.1. Subs. 1.2.2 discusses the selection of target concepts which need not be necessarily straightforward. The WordNet similarity computation `wordnetSim` is detailed in Subs. 1.2.3.

1.2.1. Mapping to WordNet

SCM takes advantage of the WordNet thesaurus to assess the similarity of a pair of concepts. WordNet groups English words into sets of synonyms called *synsets* and declares various semantic relations including hypernymy between the synsets in the form of a lexical semantic network. SCM expresses the desired set of classes as well as entities from the text as WordNet synsets. WordNet similarity measure is then used to determine the similarity between an entity and each of the classes. Soft classification is thus achieved with an entity being classified to the class with which it has the highest similarity.

Although WordNet is a comprehensive thesaurus containing approximately 146,000 word–sense pairs for nouns (as of its version 3.0), it does not contain some uncommon words and most named entities. For the purpose of resolving entities not found in WordNet, we introduced targeted hypernym discovery, which uses Wikipedia to find a hypernym to an entity.

SCM tries to map each noun phrase (entity) to a WordNet synset according to the following priorities: 1) noun phrase, 2) head noun, 3) hypernym for noun phrase, 4) hypernym for head noun. The hypernym discovery process is depicted by Alg. 2.

The match is successful if a WordNet concept with the same string representation is found. If even the hypernym for head noun is not found, the system recursively extracts a more general hypernym mappable to WordNet. The implementation of the hypernym discovery approach used is discussed in Sec. 1.3.

The extraction order was established based on small scale experimentation. Head noun is tried before the hypernym for the noun phrase, because in our experience, if the head noun is a common word present in WordNet, it is typically the type of the entire multi-word entity.

Algorithm 2 Mapping to WordNet in SCM (mapToWordNet method)

Input: *np* – noun phrase representing the entity
Output: *wordnetMap* – a WordNet mapping

```
wordnetMap:= getWordNetWord(np)
if wordnetMap = ∅ then
  head:= getHeadNoun(np)
  wordnetMap:= getWordNetWord(head)
  if wordnetMap = ∅ then
    hypernym := getHypernym(np)
    if hypernym ≠ ∅ then
      wordnetMap:= getWordNetWord(hypernym)
    end if
    if wordnetMap = ∅ then
      hypernym := getHypernym(head)
      if hypernym ≠ ∅ then
        wordnetMap:= getWordNetWord(hypernym)
      end if
    end if
  end if
end if
return wordnetMap
```

While THD for the entire noun phrase could also yield a hypernym, our impression is that the likelihood of this hypernym being incorrect is larger than if only the head noun is taken. Example 1.2 illustrates that, additionally, this hypernym may not be as specific as the head noun.

Example 1.2 (Noun phrase resolution to WordNet). Consider the noun phrase "Bucegi National Park", which has been extracted from caption of one of the images in the Czech Traveler dataset (Subs. 6.1.3). Following the priorities outlined above, the system tries to look up the following

- "Bucegi National Park" – fail. Not a WordNet concept,
- "Park" – success. "Park" is a head noun of "Bucegi National Park" and a WordNet concept.

Consider reversing the steps above: first hypernym and then head noun. The system will locate Wikipedia page "Bucegi Natural Park" and from its first sentence "The Bucegi Natural Park (Romanian: Parcul Natural Bucegi) is a protected area..." extract hypernym "protected area". Entry "protected area" is not in WordNet. The system either has to use the head noun "area", or try to get a hypernym for "protected area". The latter option would result in hypernym "location". Either "location" or "area" are inferior hypernyms (less specific) to "park". This example illustrates our reasoning behind using the head noun before trying to extract the hypernym.

The result "park" in the example above is correct, but can also be improved by syntactical analysis of the noun phrase, which will allow more informed stripping of the modifiers (i.e. try

'national park' before 'park'). Alg. 2 can be generalized to perform recursive search for best fitting hypernym. An example of such a process is given in Example 1.3.

Example 1.3 (Recursive noun phrase resolution). Consider another noun phrase, "Aughanduff", a randomly drawn title of a Wikipedia article:

- "Aughanduff" – (both noun/head noun step) fail.
- "townland" – fail not a WordNet entry. "townland" was extracted as a hypernym from the first sentence of Wikipedia article "Aughanduff": "Aughanduff is a townland in the Parish of Forkhill".[a]
- "geographical unit" – fail not a WordNet entry. "geographical unit" is a hypernym retrieved from the first sentence of Wikipedia article "Townland": "A "townland" is a small geographical unit of ..."[b],
- "unit" – success. "unit" is the head noun of "geographical unit" and a WordNet entry.

[a] Article version from November 2, 2007. THD on the current version (as of July 27, 2012) would return hypernym "hamlet", which is a WordNet entry.
[b] Article from January 14, 2008. THD on the current version (as of July 27, 2012) would return hypernym "small geographical division".

Referring to Example 1.3, it should be emphasized that Alg. 2 would have stopped at townland failing to produce mapping to WordNet. Allowing for full recursion in search for the hypernym produces a result in this example case, however, in our experience we have not observed many cases when this actually helps. Even in this example, the result – "unit" – is a rather distant hypernym for "Aushanduff".

A more severe limitation of the proposed approach is posed by the fact that many entries in WordNet have multiple synsets. For example, there are six possible meanings (synsets) for the noun 'park' as given by WordNet 3.0. The first three refer to a recreational area, the fourth one refers to the Scottish explorer Mungo Park, the fifth to 'parking lot' and the sixth to a gear position. The order of these entries is not random; WordNet actually lists the most frequently used sense of the word first. SCM uses the common baseline approach for word sense disambiguation [Agi07] and only selects the most frequently used sense. A WSD algorithm is implemented within the BOA classifier (refer to Subs. 2.4).

1.2.2. Selection of Target Classes

SCM is a semi-supervised classification algorithm, which requires (or accepts) no training instances. The classification performance is thus mainly affected by the similarity measure used and by the selection of a suitable set of target classes.

Since SCM always produces a decision, the classes should ideally cover the whole universe of entities that may appear in the input text. A theoretical option is to require some minimum similarity to classify an entity. If the similarity between the entity and the winning class is below a certain threshold, the entity is classified as "unknown". The "unknown" class would thus cover the part of the universe not covered by the classifier. Since it is not clear how to determine such thresholds, the current framework requires that the unknown class is represented by multiple specific concepts (classes).

Individual classes in the set of target classes C are designated by a human expert as a string mappable to a WordNet synset (e.g. car) or directly as a WordNet synset. The disadvantage of the former option, the one supported in our implementation, is that the mapping to a WordNet may result in multiple matching synsets.

1.2.3. WordNet Similarity Measure

The system computes the similarity between the synset representing the entity and each of the custom-defined target class concepts in C. There is a large body of work on WordNet-based measures of semantic similarity, overview of selected measures is covered by Chapter 4.

Referring to Alg. 1, the WordNet similarity is used to select the closest target concept $c \in C$ to the hypernym query mapped to $wordnetWord$:

$$class = \arg\max_{c \in C} \ wordnetSim(wordnetWord, c) \tag{1.1}$$

This equation does not define which of the senses of $wordnetWord$ should be used in the computation. As described in more detail in Chapter 4, there are generally two options. Either the most frequent sense of the word is taken with the Most Frequent Sense (MFS) strategy, or similarity values are computed for all combinations of sense assignments and the highest attained similarity value is returned with the Synset Similarity Maximization (SSM) strategy.

1.3. Targeted Hypernym Discovery

The hypernym discovery approach proposed here is based on the application of hand-crafted lexico-syntactic patterns (Hearst patterns). Although lexico-syntactic patterns have been extensively studied since the seminal work [Hea92] was published in 1992, most research have focused on the extraction of *all* word-hypernym pairs from the given generic free-text corpus. Lexico-syntactic patterns were in the past primarily used on larger text corpora with the intent to discover all word-hypernym pairs in the collection. The extracted pairs were then used e.g. for taxonomy induction [SJN06] or ontology learning [CV05a]. This effort was undermined by the relatively poor performance of lexico-syntactic patterns in the task of extracting *all* relations from a *generic* corpus. On this task, the state-of-the-art algorithm of Snow [SJN05] achieves an F-measure of 36 %.

However, applying lexico-syntactic patterns on a *suitable document* with the intent to extract *one hypernym* at a time can achieve F1 measure of 0.851 with precision 0.969 [LLM11]. In [LLM11], the suitable documents were Wikipedia entries for persons and the target of the discovery was the hypernym for the person covered by the article.

We introduce name "Targeted Hypernym Discovery" (THD) for this approach. The goal of THD is not to find all hypernyms in the corpus but rather to find hypernyms for a specific entity. The THD algorithm proposed here is an updated and expanded version of the algorithm used in our earlier work [CKN+08]. The algorithm requires no training and can use up-to-date on-line resources to find hypernyms in real time. The outline of the steps taken to find a hypernym for a given entity in our THD implementation is denoted in Alg. 3.

Performing Alg. 3 requires to carry out multiple information retrieval and text processing tasks. For text processing, our THD implementation uses the GATE NLP Framework

Algorithm 3 Targeted Hypernym Discovery (getHypernym procedure)

Input: *np* – noun phrase representing the entity, *maxArticles*
Output: *hypernym* – a hypernym for the entity

```
//if there are only n matching articles, n < maxArticles articles, get all
doc[] := get top maxArticles documents matching np
for i:=1 to |doc| do
  if doc[i] matches np then
    hypernym := extractHypernym(doc[i])
    if hypernym ≠ ∅ then
      return hypernym
    end if
  end if
end for
return ∅
```

[CMBT02]. In Subs. 1.3.1, we substantiate the choice of Wikipedia as the corpus and explain the way it is interfaced and the documents are preprocessed. Information retrieval from Wikipedia is covered in Subs. 1.3.2. The text processing symbolically denoted by the *extractHypernym* function call in Alg. 3 is covered in Subs. 1.3.3.

1.3.1. Wikipedia as the Corpus

A gold-standard dataset for training and testing hypernym discovery algorithms is WordNet (e.g. [SJN05, SJN06]). WordNet's structured nature and general coverage make it also a favourite choice for general disambiguation tasks. However, the frequent occurrence of named entities in some datasets makes the use of WordNet and probably most other closed lexical resources infeasible. This is documented in the study [SJN05], which evaluated several hypernym discovery algorithms on a hand-labeled testset where over 60% of entities were named entities. The performance of the best algorithm based on lexico-syntactic patterns significantly surpassed the best WordNet-based classifier (F-Measure increase from 0.2339 to 0.3592).

The goal of THD is to improve the coverage of SCM by mapping entities that do not occur in WordNet to WordNet synsets through hypernyms extracted from a suitable large free-text corpus. We opted for Wikipedia, the fast growing, publicly available encyclopedia.

Unlike [BC07] who combined web search, Wikipedia article titles and hyperlinks for extraction of instances of arbitrary relations or [SKW07] who mainly use the Wikipedia category system for the purpose of ontology learning, we found the first section of Wikipedia articles as particularly suitable for hypernym discovery and use it as the sole source of information. More details on the suitability of Wikipedia for hypernym discovery are present in Chapter 5.

1.3.2. Wikipedia Search

The selection of articles for hypernym discovery is the main differentiator between the system presented here and other approaches in the literature. For a given entity (query for hypernym), an online search in English Wikipedia is executed through the Wikipedia Search API, which

provides access to Wikipedia's Lucene-based fulltext search.[1]

Example 1.4 (Search result for query "Maradona").

```
<api>
<query>
<searchinfo totalhits="804"/>
<search>
<p ns="0" title="Diego Maradona" snippet="Diego Armando <span
class='searchmatch'>Maradona</span> (...; born 30 October 1960) is a former
Argentine football player.
He is widely regarded as one of <b>...</b> " size="69655" wordcount="9140"
timestamp="2011-05-17T07:43:37Z"/>
<p ns="0" title="Hugo Maradona" snippet="Hugo Hernán <span
class='searchmatch'>Maradona</span> (born May 9, 1969), also known as
El Turco, is an Argentine Association football coach and former player.
<b>...</b> " size="6075" wordcount="692" timestamp="2011-04-08T09:30:47Z"/>
<p ns="0" title="Maradona by Kusturica" snippet="<span
class='searchmatch'>Maradona</span> by Kusturica is a documentary on the life
of Argentine footballer Diego <span class='searchmatch'>Maradona</span> ,
directed by the award-winning Serbian <b>...</b> " size="3267" wordcount="344"
timestamp="2011-04-23T05:31:07Z"/>
...
</search>
</query>
<query-continue>
<search sroffset="10"/>
</query-continue>
</api>
```

Search url: `http://en.wikipedia.org/w/api.php?action=query&format=xml&list=search&srwhat=text&srlimit=10&srsearch=maradona` [Retrieved on April 4, 2011]

Articles are ranked according to textual similarity and also on the number of in-links they receive. This ensures that e.g. for query "Gates" the first article in the search result list is an article on "Bill Gates", and not an article on some other person named Gates, which would have probably be produced on the basis of pure textual match. Our assumption is that the higher the article in search results the higher the probability that the article is a correct match for the query. Since in our current work we stick to this "most frequent sense assumption", the articles are processed in the order of their appearance in the search results. Example 1.4 gives a sample search result.

The full texts of a Wikipedia article is obtained via the `Special:Export` interface of Wikipedia's MediaWiki engine[2], which puts less strain on Wikipedia's resources than obtaining a Wikipedia article. A sample export via this interface is given by Example 1.5.

[1] `http://www.mediawiki.org/wiki/Extension:Lucene-search` [Retrieved on June 11, 2012]
[2] `http://www.mediawiki.org`

Example 1.5 (Article "Diego_Maradona" retrieved via Special:Export).

```
...
<page>
<title>Diego Maradona</title>
<id>8485</id>
<revision>
<id>429522687</id>
<timestamp>2011-05-17T07:43:37Z</timestamp>
<contributor> <username>Gelu6</username>
<id>13663249</id>
</contributor>
<text xml:space="preserve" bytes="69655">
{{Use dmy dates|date=May 2011}}
{{Infobox Football biography 2
| playername = Diego Maradona
| image = [[Image:Diego Maradona.jpg|265px]]
| fullname = Diego Armando Maradona
...}}
'''Diego Armando Maradona''' (...; born 30 October 1960) is a former [[football
in Argentina|Argentine football]] player.  He is widely regarded as one of the
greatest..  </text>
</revision>
</page>
</mediawiki>

Search url:  http://en.wikipedia.org/wiki/Special:Export/Diego_Maradona, [Retrieved
on May 17, 2011]
```

Filtering Search Results In many cases, the article title is spelled differently or with a word missing or added as compared to the query. Extracting hypernyms from articles that only loosely match the query would deteriorate the performance of the system; it is therefore necessary to determine if the article is on the entity in the hypernym query. In order to make this decision we compute a string similarity between the article title and the query. Our system uses the Jaro-Winkler similarity, since this measure was specifically developed for matching named entities (people names) [WT91]. If this similarity is below a given threshold, the article is excluded from further processing.

Since the capitalization of the query and the article topic should match, but Wikipedia capitalizes all the article headings, the article text is previewed to see if the topic of the article tends to appear in upper case or lower case in article body. This for example removes an article titled "Logical Gates" from the search result for query "Gates", the reason is that the word "Gates" appears in the article "Logical Gates" mostly in lower case.

Obtaining full text The full texts of a predefined number of top ranked Wikipedia articles that passed the selection outlined above are obtained through the `Special:Export` interface of Wikipedia's MediaWiki engine described earlier. Since the first section of the article is generally most informative, the remaining sections are dropped. Also, wiki-markup, links,

hidden text such as comments, information boxes etc. are stripped.

Our system interfaces with Wikipedia through a newly-designed `WikipediaPR` module, which is described in Subs. 1.5.2.

1.3.3. Pattern Matching

The input for this phase is a "hypernym query", an unknown entity, and a Wikipedia article that is assumed to cover the unknown entity. The output is a hypernym. According to our experimental evaluation, the first section of an article provides a sufficient basis for THD since it contains a brief introduction to the topic of the article, often including the desired definition in the form of a Hearst pattern. Processing the remaining sections, in our experience, generally only increases the computation time and introduces incorrect hypernyms.

In GATE framework, the linguistic information is attached to the underlying text using layers of *annotations*. The annotations are assigned additional pieces of information using *features*. For the purpose of the THD grammar detailed in this subsection, there are two important annotation types `Token` and `Highlight`. For `Token` annotation, the grammar works with the following features:

- `string`: underlying text
- `category`: POS tag
- `split`: true if the underlying text is a sentence split, otherwise false.

The `Highlight` annotation is placed over occurrences of the hypernym query. It is processed twice, first by the grammar without any features, and then including features in the post-processing phase; the description of features of `Highlight` annotation is therefore postponed into Subs. 1.3.4, which covers the post-processing.

The `NounChunk` annotation is not used directly by the grammar, but it is used from the application to extract a more precise hypernym from the input text. JAPE grammars (Java Annotation Patterns Engine) [CMT00] are used to create and execute Hearst patterns over these annotations which extract the hypernym.

JAPE grammars The NLP components that perform text preprocessing in the GATE framework append their output to the existing text in the form of annotations. JAPE provides a finite-state transduction over these annotations.

A JAPE grammar [CMT00] is processed by a JAPE transducer, a GATE PR module. Input for a JAPE transducer is a JAPE grammar, which is basically a set of rules, and a preprocessed text to annotate. JAPE rules consist of left- and right-hand side. On the left-hand side, there is a regular expression over existing annotations; annotation manipulation statements are on the right-hand side. A JAPE grammar was already used to match Hearst patterns in [CV05a]. However, their paper does not elaborate on the grammar used or on its performance in detail.

JAPE grammar for Hearst Patterns Alg. 4 gives a sample JAPE grammar used for hypernym discovery in THD, for full listing refer to Appendix A. The grammar was designed for speed. Although the use of the {1,} and {0,} repeat operators "+" and "*" would simplify and generalize the macro, it would also have deteriorating impact on the processing speed[3], which

[3] According to GATE documentation [CMB+12]: "Optimising for speed: ... avoid the use of the * and + operators. Replace them with range queries where possible."

we wanted to avoid. Other lexico-syntactic patterns identified by Hearst [Hea92], e.g. the 'such as' pattern, were not considered, because they did not seem to provide a significant improvement from our preliminary observation. To illustrate the grammar in action, we show in Alg. 4 how it would extract the hypernym for "Maradona" from the following sentence:

Diego Armando Maradona (born 30 October 1960) is a former Argentine football player.

Comments delimited by '//' are used to align the matched text with individual subpatterns of the grammar. The grammar references the `Token` and `Highlight` annotation described above and three *macros*: `NameOf`, `LHSHearstBody` and `Head`. From the point of the embedding grammar, a macro can be viewed as an annotation. These macros are detailed below.

Algorithm 4 Sample JAPE grammar for extracting Hearst patterns

Input: Tokenized text annotated with QueryHighlightPR

Output: The input extended with the hypernym marked with hearstPattern annotation or the input if the rule did not fire

```
// rule matches patterns only within one sentence
Rule: ExampleHearstPattern
Priority:1000
// rule-specific macro to match the query
(Highlight) //matches 'Maradona'
// matches any number of any tokens
({Token.split="false"})* // matches '(born October 30, 1960)'
// followed by a form of 'to be', here 'is'
{Token.string == "is"}|{Token.string == "are"}|{Token.string == "were"}|{Token.string
== "was"}
// followed by an article
({Token.string == "a"}|{Token.string == "an"}|{Token.string == "the"}) //matches 'a'
(NameOf)? // does not match anything in this example
// followed by macro a defining allowed words preceding the actual hypernym for query
(LHSHearstBody) // matches 'former Argentine football'
// hypernym can be only NN, NNS or NNP
(Head) // matches 'player'
:hearstPattern
 ⟹ // delimits LHS from RHS
// new 'hearst' annotation is added to 'player' the string identified by the hearstPattern
label
:hearstPattern.hearst = {rule = "Example"}
```

NameOf

The `NameOf` macro (matching string "name of") accommodates for cases, when there is "is a name of" filler inserted between the verb and the hypernym in the sentence as in:

Horsfieldia is a name of plant genus native to South East Asia *First sentence of the Wikipedia article 'Horsfieldia' as of 6 April 2011.*

The complete macro can be found in Appendix A.

LHSHearstBody

The `LHSHearstBody` macro matches the following pattern:

Listing 1.1: LHSHearstBody macro

```
Token? CD? JJ? JJ? NNP? NNP? VBN? JJ? JJ? NN? NN? NN? NN?
```

`Token?` matches any single token (word, comma etc.) or nothing, `CD` matches a cardinal number, `JJ` an adjective, `NNP` a proper noun, `NN` a noun and `VBN` a verb. The complete grammar can be found in Appendix A.

Head

The `Head` macro matches the following pattern:[4]

Listing 1.2: Head macro

```
NN|NNP|NNS
```

This captures the head noun of the hypernym ("player" in the example). In many cases, the noun phrase embedding the head makes for a better and more specific hypernym.

Linguistic Processing Outside JAPE grammar

THD has two outcomes: the (proper) noun annotated with the 'hearst' annotation ('player') and the noun phrase in which it is contained ('former Argentine football player'). This noun phrase can be in some cases identical with the noun, but ideally it should provide a less general hypernym for the query. To obtain the embedding noun phrase, a noun chunker is run as part of the preprocessing. After the JAPE grammar is executed, the noun phrase embedding the head noun is extracted and marked as a second outcome of the algorithm. The grammar also transfers features from the `Highlight` annotation to the resulting `hearst` annotation. These features are then used by the hypernym filtering process, which is covered by Subs. 1.3.4.

1.3.4. Filtering Hypernyms

As shown in the full JAPE listing (Alg. 17, Appendix A), the features on the `Highlight` annotation are transfered to the `hearst` annotation, which is created on the hypernym. These features are used in a subsequent hypernym filtering step, where hypernyms not matching externally set constraints are omitted.

The annotation has four features describing the type of the match produced:

- `Type` (query/title/article_start)
- `Full` (true/false), value *false* indicates partial match against the query or article title

[4] The "|" symbol denotes logical or.

- `DiacriticsMatches` (true/false) diacritics of the query or title matches the string in the document on which the annotation is placed,
- `CaseMatches` (true/false) the casing of the query or title matches

In general, the *true* values for the `Full`, `DiacriticsMatches` and `CaseMatches` features increase precision.

The rationale for the `Type` parameter is that, in our observation, the full name of the entity as reflected also in the name of the article tends to appear in the first sentence of the article rather than the (often abbreviated) hypernym query.

Example 1.6 (Matching article title instead of hypernym query). For hypernym query "Maradona" the corresponding article is named "Diego Maradona" and the sentence featuring the Hearst pattern starts "Diego Armando Maradona ... is a " (refer to Example 1.4). The query is better matched with the article text than article title in this case. However, if the hypernym query is Maradona's nickname "El Pibe de Oro" (Spanish for "the golden kid") then the document title is more suitable to be matched against the article text than the query.

If the match is full or partial is determined using regular expressions over plain text. The sought string (hypernym query or title) is broken down into words and a regular expression is generated from it. For each match of the regular expression it is assessed whether the match spans all the sought words or not.

We also experimented with matching the article start to cover articles, which have first sentence containing the hypernym as object, but the subject is missing (refer to Appendix A). Since it appears that there is a very small number of such articles, this annotation type is not practically important.

Hypernym filtering is technically performed by the `WordnetSimilarityPR` described in Subs. 1.5.1. The `Highlight` annotation is created by the `QueryHighlightPR` described in Subs. 1.5.2.

1.4. Example

Consider the picture of a footballer scoring a goal, which is assigned the textual annotation "Maradona hits the net again". This hypothetical classifier assigns the following labels (classes): {*hockey player, footballer, basketball player, swimmer, runner, sports equipment*}. In order to aid this classifier in its uneasy task, SCM can be used to determine which of the classes probably appear in the image based on the textual annotation.

The annotation is broken down into two entities, 'Maradona' and 'net', and then the system attempts to map each of these entities to WordNet. Following the steps described in Subs. 1.2.1, SCM tries to find an entry for 'Maradona' in WordNet, but there is no such entry. The system calls the THD algorithm to return a hypernym for 'Maradona'. THD finds a Wikipedia entry entitled 'Maradona', and with a Hearst pattern it extracts the hypernym 'football player'. The noun phrase 'Maradona' is then matched with a synset that has 'football player' attached. Finally, the system computes the similarity between this synset and the synsets representing each of the target classes using a WordNet similarity measure: class 'footballer' is correctly assigned the highest confidence; it has similarity 1 because it belongs to the same WordNet synset as 'football player'.

concept/entity	Maradona	net
footballer	**1.0**	0.11
sports_equipment	0.126	**0.60**
hockey_player	0.720	0.11
runner	0.705	0.34
swimmer	0.687	0.10
basketball player	0.710	0.11

Table 1.1.: Results of SCM on "Maradona hits the net". The WordNet similarity measure used is Lin (refer to Subs. 4.2.4).

SCM then proceeds to the second entity, 'net'. THD is not used, because multiple synsets described with this word are found directly in WordNet. Noun 'net' is then mapped, using the MFS strategy, to its first, most frequently used meaning/synset, which is the 'the computer network' sense. Luckily, the sports equipment target class is, nevertheless, assigned the highest similarity (0.60).

The result obtained with our implementation is displayed on Table 1.1. The setup was as follows: Lin similarity measure from the JWNL similarity library and using the information content values computed over the British National Corpus with Resnik counting with smoothing (refer to Subs. 4.3.1).

1.5. Implementation

This section gives details on standalone modules (GATE Processing Resources), which were developed within the scope of this dissertation for the purpose of targeted hypernym discovery. An overview of the workflow is given by Fig. 1.1. The input text is parsed for entities with the standard GATE modules. SCM, including THD, is implemented in Java on top of the GATE framework. The core function of SCM – the similarity computation – is done by `WordNetSimilarityPR`. Before it is run, `Token` and optionally `NounChunk` annotations must be created on the input text. `WordNetSimilarityPR` reads the input GATE annotations and writes its output also in the form of GATE annotations. Optionally, the `stem` feature on the `Token` annotation is produced by Snowball stemmer (`http://snowball.tartarus.org`). It is then used in `WordnetPR` for matching the entity with WordNet instead of the original text.

At this step, the application either preserves or drops casing depending on the value of the `requireCaseMatch` boolean feature. For example, if "Bath" is stemmed to "bath", it is reverted to "Bath".

1.5.1. WordNetSimilarityPR

The core SCM functionality is implemented by the `WordNetSimilarityPR`. This PR goes through the input text and tries to classify all the matching entities with a selected WordNet similarity measure as was described in Subs. 1.2.3. If the entity is not found in WordNet, THD is initiated according to Alg. 2.

Our implementation offers a choice of any single similarity measure implemented in the JWordnetSim library or the JWSL library, both detailed in Sec. 4.3. Based on experiments

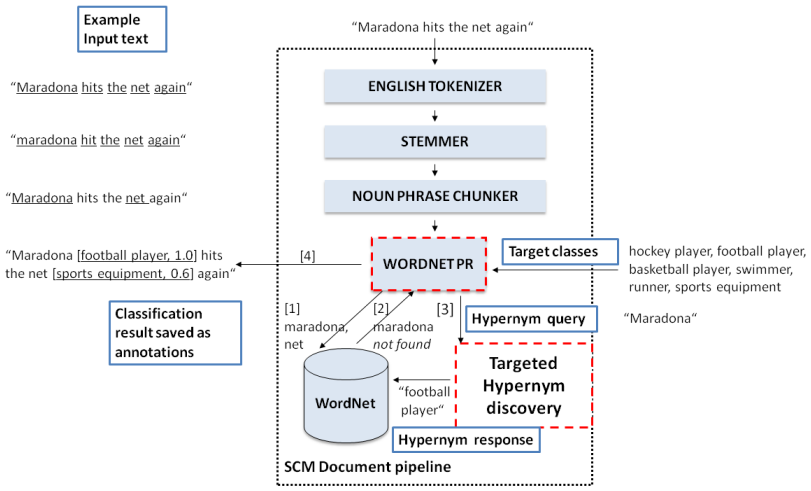

Figure 1.1.: Semantic Concept Mapping Workflow. Dashed boxes indicate contributions within this dissertation

presented in Sec. 6.2, the Lin measure, supported in both libraries, was selected as the default one. Supported sense selection strategies within SCM are MFS for JWNL and SSM for JWSL.

Features

- `useNounChunks` (boolean): input annotations are `Token` (false) or `NounChunk` annotations covering noun phrases (true).
- `useWikipedia` (boolean): if true, THD is initiated if word not resolved in WordNet

If `useWikipedia` is set to true, the values of these additional features are used: `requireCaseMatch`, `requireDiacMatch`, `wikiEntriestoTry`, `allowTitleMatch`, `articleTitleMatchThreshold`, `allowArticleStartMatch`, `requireFullMatch`.

Matching against WordNet The system tries to map the annotation to WordNet. If not successful and the current annotation is a noun phrase, SCM repeats the WordNet matching attempt with the head noun of the current annotation. If that also fails it resorts to THD according to Alg. 2.

1.5.2. Targeted Hypernym Discovery

Running THD within GATE requires two pipelines: the corpus acquisition pipeline and the corpus annotation pipeline with Wikipedia articles (see Fig. 1.2).

The *corpus acquisition pipeline* contains only the `WikipediaPR`, which populates the corpus for a given hypernym query.

The *corpus annotation pipeline* uses predominantly existing GATE modules to perform text preprocessing. The only exception is the `QueryHighlightPR`, used for highlighting the hypernym query in the text, which is the only contributed PR to this pipeline. Noun phrases

are identified using Ramshaw-and-Marcus noun phrase chunker [RM95] available in GATE. Other modules come from the GATE reference information retrieval and extraction system ANNIE (a Nearly-New Information Extraction System). The `JAPE Transducer PR` performs the THD business logic: it is used by the THD with the grammar described in Sec. 3.2 to discover Hearst patterns in the text.

In the typical case, multiple hypernyms might be extracted for each hypernym query, both within one document and in multiple documents in the corpus. The selection of the correct hypernym is done by `WordNetSimilarityPR` as described in Subs. 1.3.4.

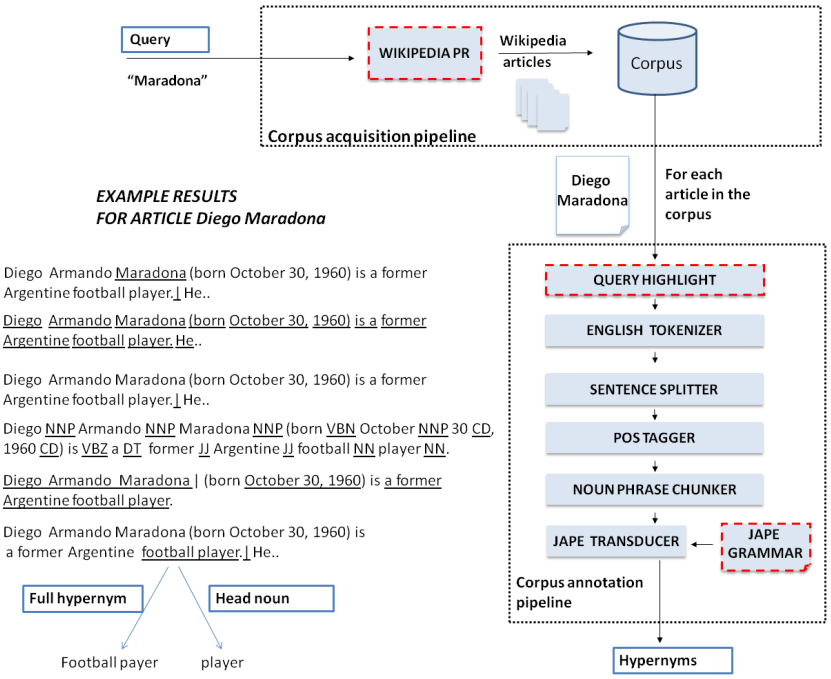

Figure 1.2.: Targeted Hypernym Discovery Workflow. Dashed boxes indicate contributions within this dissertation

WikipediaPR

This processing resource implements the functionality described in Subs. 1.3.2. It populates a GATE corpus with articles relevant to the `query` parameter.

The parameters (features) for this PR are:

- `query` (string): the hypernym query
- `limit` (integer): maximum number of articles to retrieve. The number of retrieved articles may be smaller.
- `ArticleTitleMatchThreshold` (float): minimum Jaro-Winkler similarity between article title and query. The default value is 0.9.

- `stripDiacriticsForTitleComparison` (boolean): if set to true the similarity comparison between query and article title is performed on the diacritics-stripped versions of the respective strings.

The query is saved as a document feature `query`. While most of the Wiki markup is not processed and directly stripped, the section start information (denoted by "==section name==" markup) is used to mark the heading of the article and each section.

QueryHighlightPR

`QueryHighlightPR` generates the `Highlight` annotations as described in Subs. 1.3.4. The text to be highlighted is taken either from the query parameter of the PR, or from the "query" feature on the document processed (the former has a priority if both are present). No input annotations or preprocessing is required. The way `QueryHighlightPR` works is detailed in Alg. 5.

Algorithm 5 Highlighting query in the input text (QueryHighlightPR processing resource)
Input: *textToHighlight*, *document* – string representation of the document
Output: *highlightAnnotations*[] – annotations over textToHighlight in the document

highlightAnnotations[] := {∅}
strippedContent :=strip(*document*)
//the strip function removes diacritics
regex := buildRegexCaseInsensitive(breakToWords(strip(*textToHighlight*)))
for all *match* in matches(*regex*,*document*) **do**
  annotation := newAnnotation(*match*)
  if length(*match*) = length(*textToHighlight*) **then**
    $annotation.features$= $annotation.features$ ∪ "Type=full"
  else
    $annotation.features$= $annotation.features$ ∪ "Type=partial"
  end if
  if diacriticsMatches(*match*,*textToHighlight*) **then**
    $annotation.features$= $annotation.features$ ∪ "DiacriticsMatches=true"
  else
    $annotation.features$= $annotation.features$ ∪ "DiacriticsMatches=false"
  end if
  if caseMatches(*match*,*textToHighlight*) **then**
    $annotation.features$= $annotation.features$ ∪ "CaseMatches=true"
  else
    $annotation.features$= $annotation.features$ ∪ "CaseMatches=false"
  end if
  highlightAnnotations[]:=*highlightAnnotations* ∪ *annotation*
end for
return *highlightAnnotations*[]

1.6. Contribution

The SCM algorithm is based on "off-the-shelf" WordNet similarity measures. The performance of this algorithm on the standard WordSim353 dataset is presented in Sec. 6.2. The result, Spearman correlation coefficient around 0.33, is mainly determined by the performance of the underlying WordNet measures, as almost all entries in the dataset are words covered by WordNet. The performance is rather substandard compared to results of state-of-the art Wikipedia-based algorithms.

The application of SCM on the Czech Traveler dataset is covered by Sec. 6.3. The THD algorithm correctly discovers a hypernym in 62% of the 47 entities in the dataset, which could not otherwise be mapped to WordNet. Overall, the SCM produced a correct result for 74% of entities with ground-truth available.

The Wikipedia-based THD results are encouraging, although further experiments on a larger and more representative sample are needed. Apart from the free text, there is also semistructured information contained in Wikipedia articles – the infoboxes and the categories to which the article is assigned. This prospective source of hypernyms is left unexploited by our THD algorithm and can lead to additional improvement.

An interesting property of both SCM and THD is that no training data are required. THD can be perceived to be an *unsupervised learning* algorithm, since there is no user-defined set of target classes. The classes (types of entities) are discovered by text mining/NLP techniques from Wikipedia.

We consider the main contribution of this chapter to be the workflow for hypernym discovery from Wikipedia to WordNet entries. This allows to apply Wordnet similarity measures on entities not covered by WordNet. Overall, the design, implementation and experiments with the SCM algorithm pointed at the strong points and limitations of Wikipedia and WordNet and as such served as a starting point for the BOA algorithm introduced in the next chapter.

2. Bag-of-Articles Classifier

This chapter introduces our attempt for a fully Wikipedia-based entity classification algorithm. The approach presented here builds upon the encouraging result obtained with Wikipedia on the hypernym discovery task presented in the previous chapter, and extends the use of Wikipedia also to classification, where it replaces WordNet used in the SCM algorithm. Unlike WordNet, where most information about similarity of two synsets is contained in the path in the thesaurus connecting them while the textual descriptions are very short, Wikipedia is less rigidly organized, but contains in average about 300 words per article.[1] By representing the noun phrase and the class as Wikipedia articles, then the noun phrase classification translates to the task of measuring semantic relatedness of two documents.

As will be shown in Subs. 3.3.1, researchers report that using the text of Wikipedia articles as an input for a bag-of-words classifier does not yield good results in the word similarity computation task. The Bag-of-Articles (BOA) approach presented here builds upon the bag-of-words approach but utilizes a richer, more robust and more elaborate feature set. It consists of the BOA *classifier* that operates on top of the BOA *representation*.

BOA representation The target class and the unlabeled instance are considered *entities*. The steps to create a BOA representation for an entity are:

- **MAPPING:** entity is mapped to one or more entity articles,
- **CRAWLING:** entity articles are seeds for crawling which gathers additional articles,
- **AGGREGATION:** all articles are aggregated into a constant-length term vector.

BOA classifier is effectively a Rocchio classifier (refer to Subs. 3.5.1), which uses BOA representation for both the target class and the unlabeled instance. In the Rocchio classifier, only the target classes are created from multiple documents, while the unlabeled instance is a single document. In other respects, the BOA classifier is only a concretization of the Rocchio classifier: normalization is used, there is a specific choice of term-weighting function and negative instances are not used.

The main contribution of this approach is thus the way the BOA representation is created. In the BOA representation the weight vector for both target class as well as for the unlabeled instance is created from multiple hyper-linked documents. These articles are added according to a weighting scheme, which takes into consideration the "link type" (we call this *modality*), the distance of the article from the seed and supports multiple term-weighting functions, including a novel WordNet-based one.

This chapter is organized as follows. Sec. 2.1 introduces the BOA classifier and Sec. 2.2 the BOA representation. A detailed description of individual parts of the algorithm is in Sec. 2.3. An entity disambiguation algorithm using BOA is presented in Sec. 2.4. Sec. 2.5 demonstrates

[1] English Wikipedia 2010; `http://en.wikipedia.org/wiki/Wikipedia:Size_comparisons`

the BOA approach on a toy example. Sec. 2.6 our BOA implementation. A genetic algorithm for learning the weights of the externally set parameters of the BOA algorithm is presented in Sec. 2.7. The contribution of the BOA algorithm in the field of Wikipedia-based WSC is outlined in Sec. 2.8 along with directions of prospective future work. Table 2.9 located in the end of the chapter gives an overview of the notation used.

2.1. BOA Classifier

A BOA classifier is trained for a set of target classes $c \in C$, each represented by one or more training instances – Wikipedia articles. These articles are used as seeds for gathering related articles. For each target class all the articles – the training instances + automatically gathered related articles – are aggregated into one pseudo-document, which is represented using the Bag-of-Words (BOW) model. We call this pseudo-document a BOA representation.

The unlabeled instances are represented as noun phrases. For these instances, the BOA classifier locates articles in Wikipedia that might define the entity and selects one of them using a disambiguation function. Subsequently, it proceeds in the same manner as in training – related articles are identified, aggregated into a pseudo-document which is again represented as a BOW term-weight vector.

The classification step is very simple – the class is assigned by comparing the instance's BOA representation with BOA representation of each target class with selected similarity measure. The most similar target class is the result of the classification. In the rest of this section, we will introduce the basic workflow of the BOA classifier with a formal notation,[2] which is used throughout the chapter.

2.1.1. Source Data

We use symbol $\mathcal{A}_{wiki}$ to denote a collection of all articles in a given Wikipedia snapshot. Each article is described by its title, term-weight vector, and modality membership indices: out-links, in-links and a list of categories the article belongs to. The BOA representation, as proposed here, does not process Wikipedia infoboxes and ignores wiki and HTML markup.

2.1.2. Training Phase

The input of a training phase of a BOA classifier is a set of target classes, each described by a name and an optional disambiguation by a WordNet synset. A target class $c \in C$ is represented as:

$$c = \langle \text{class name}, \mathcal{A}_c, \mathcal{NP}_c (,\text{WordNet synset}) \rangle, \tag{2.1}$$

where $\mathcal{A}_c \subset \mathcal{A}_{wiki}$ is a set of training instances – seed articles classified as c, $\mathcal{NP}_c$ is a set of noun phrases given for c, and the optional WordNet synset is used by some term-weighting functions. At least one of the sets $\mathcal{A}_c$ and $\mathcal{NP}_c$ must not be empty. In a typical training setting, $\mathcal{A}_c$ contains one or more elements and $\mathcal{NP}_c$ is empty.

If $\mathcal{NP}_c$ is not empty then the BOA mapping step $\rightarrow_{map}$ selects for each $np \in \mathcal{NP}_c$ the closest Wikipedia article (for more details refer to Subs 2.2.1). The result of the mapping of

[2] Generally, the notation was created with these guidelines in mind: capital letters are used for constants and matrices, sets are typed in calligraphic font, small Greek letters are used for functions and small letters for variables. To aid readability, multi-letter subscripts and occasionally superscripts are used.

noun phrases is put to $\bar{\mathcal{A}}_c$ along with the set $\mathcal{A}_c$, which contains training instances already mapped to Wikipedia articles (also refer to Eq. (2.9)).

The set of all *entity articles* is denoted as:

$$\mathcal{A}_{train} = \bigcup_{c \in C} \bar{\mathcal{A}}_c. \tag{2.2}$$

We also define $\hat{\mathcal{A}}_{train}, \hat{\mathcal{A}}_{train} \supseteq \mathcal{A}_{train}$ as a set of all articles *involved* in training – see also Eq. (2.18). The set $\hat{\mathcal{A}}_{train}$ includes in addition to entity articles also the gathered related articles.

The set $\hat{\mathcal{A}}_{train}$ is an input for the term selection function σ, which defines the vector space used by the classifier in terms of its dimensionality and the correspondence of individual components with specific terms.

$$\sigma(\hat{\mathcal{A}}_{train}) \rightarrow \mathcal{T}^N, \tag{2.3}$$

where $\mathcal{T}$ is the set of all terms. Note that this set may include also terms not present[3] in any of the articles in $\mathcal{A}_{wiki}$.

The bag-of-articles function β associates the target class c with an N-dimensional vector of term weights $\langle w_1, \ldots, w_i, \ldots, w_N \rangle$. The i-th component of the vector corresponds to a weight of the i-th term in the range of $\sigma(\hat{\mathcal{A}}_{train})$:

$$\beta(c) \rightarrow \mathcal{R}^N. \tag{2.4}$$

The output of a training phase is a BOA classifier – term-weight vector $\langle w_1, \ldots, w_N \rangle$ for each of the target classes $c \in C$.

2.1.3. Classification Phase

The unlabeled instance x formally closely resembles training instance, only the WordNet synset is not present:

$$x = \langle \text{instance name}, \mathcal{A}_x, \mathcal{NP}_x \rangle, \tag{2.5}$$

where instance name is a noun phrase, $\mathcal{A}_x \subset \mathcal{A}_{wiki}$ is a set of known Wikipedia articles on the classified entity and $\mathcal{NP}_x$ is a set of noun phrases given for x. In a typical classification setting, the instance name will be a noun phrase np extracted from text and the set $\mathcal{A}_x$ will be empty and $\mathcal{NP}_x$ will contain one element – the noun phrase np, which will be mapped to an entity article and placed to $\bar{\mathcal{A}}_x$. Formally, the same applies to sets $\mathcal{A}_x$ and $\mathcal{NP}_x$ as described in Subs. 2.1.2 for $\mathcal{A}_c$ and $\mathcal{NP}_c$, including the mapping to $\bar{\mathcal{A}}_x$ according to Eq. (2.9).

The BOA function β projects the unlabeled instance (mapped to a set of seed articles $\mathcal{A}_x$) onto an N-dimensional vector of term weights:

$$\beta(x) \rightarrow \langle w_1, \ldots, w_N \rangle. \tag{2.6}$$

The dimensionality of the vector is the same as in the training phase. All the additional terms that are run into in a classification phase, but did not occur in the training phase, are ignored. Both a disambiguated unlabeled instance x and target classes $c \in C$ are represented with values of the bag-of-articles function $\beta \rightarrow R^N$. The most similar class is assigned simply as:

[3] Additional terms may be introduced as a consequence of lemmatization (refer to Subs. 2.3.2).

$$class(x) = \arg\max_{c \in C} sim(\beta(c), \beta(x)), \tag{2.7}$$

where *sim* refers to selected similarity measure and **arg max** stands for the argument of the maximum, which is a set of points of the given argument for which the given function attains its maximum value. Here, this function selects the class with maximum similarity with the unlabeled entity. If there are multiple such classes and the use of the BOA classifier requires a single class, one can be selected arbitrarily. It should be noted that for real world scenarios, it is unlikely that two classes attain the same non-zero maximum value.

2.2. BOA Representation – the Three Steps of BOA

The BOA representation is created in three steps: mapping, crawling and aggregation. The result of this process is a BOA representation – a vector of term weights. The BOA representation is the well-known (BOW) model applied in the context of hyper-linked article structure.

For the purpose of BOA the target classes and unlabeled instances are considered as entities and dealt with in the same way:

$$e = \langle \text{entity name}, \mathcal{A}_e, \mathcal{NP}_e \rangle. \tag{2.8}$$

We will continue the discourse choosing the symbol e (for entity) as a superconcept for both the unlabeled instance x and the target class c.

2.2.1. Mapping

This step accounts for mapping the entity to one or more *entity pages* – Wikipedia articles. The input for the mapping step are sets $\mathcal{A}_e$ and $\mathcal{NP}_e$. The result of mapping is the set $\bar{\mathcal{A}}_e$. Symbolically:

$$\bar{\mathcal{A}}_e = \mathcal{A}_e \bigcup_{np \in \mathcal{NP}_e} (np \rightarrow_{map} \mathcal{A}_{wiki}). \tag{2.9}$$

If the set of noun phrases $\mathcal{NP}_e$ defined for entity e is empty, the mapping step is trivial $\bar{\mathcal{A}}_e = \mathcal{A}_e$. If $\mathcal{NP}_e$ is not empty, the $np \rightarrow_{map} \mathcal{A}_{wiki}$ denotes a result of subsequent application of ranking function ρ and disambiguation function δ:

$$\vec{s} = \rho(np) \tag{2.10}$$

$$a = \delta(\vec{s}), a \in A_{wiki}. \tag{2.11}$$

The first step is to determine the candidate articles from $\mathcal{A}_{wiki}$ that are considered to correspond to the various senses of each of the noun phrases $np \in \mathcal{NP}_e$. To this end serves the ranking function ρ, which is a vector-valued function that associates the noun phrase np with the vector of its n possible senses:

$$\rho(np) \rightarrow A^n_{wiki}, \tag{2.12}$$

where n is the number of possible senses of np.

The senses – *Wikipedia articles*[4] – are sorted in the vector in the decreasing order of esti-

[4] Technically, in a given Wikipedia snapshot an article is unanimously identified by its title.

mated probability of being the correct sense for the instance in a random context. The sense s_1 of the noun phrase np is represented by the *entity article* $a_{s_1} \in \mathcal{A}_{wiki}$. The disambiguation function $\delta(\vec{s})$ chooses one entity article from the set of the senses. In the base scenario, we use disambiguation function δ_{mfs}, which assigns the **m**ost **f**requent **s**ense:

$$\delta_{mfs}(\vec{s}) = a_1, \tag{2.13}$$

where s_1 refers to the first component in the input vector returned by $\rho(np)$. It follows from the properties of the ranking function ρ that the first component is the most frequent sense. For a more elaborate disambiguation algorithm please refer to Sec. 2.4.

2.2.2. Crawling

Entity articles retrieved in the mapping step are used as seeds in a crawling task with the purpose to obtain multiple additional articles which are closely topically related to the seed entity article.

Articles are gathered using one or more *modalities*. The modality is crawled starting from the seed in the direction of the edges. The stopping criterion is reaching a path of certain length. The set of modalities is defined separately for training phase as $\mathcal{M}_{train}$ and classification phase as $\mathcal{M}_{test}$ and the crawling depth is defined independently for each modality and phase.

To illustrate the notion of modality, refer to the example below.

Example 2.1 (Modality functions). Using the *out-link* as a modality we can view the Wikipedia as a directed cyclic graph of article pages. There is an edge between two pages a_i and a_j if there is a hyperlink from a_i to a_j. A special example of a modality is the *same category* modality. In this modality, the Wikipedia graph will have an edge from a_i to a_j and from a_j to a_i if a_i and a_j share the same category, e.g. football. The last example – *in-link* modality. Here, the graph looks similar as in the out-link example, but the direction of the edges is reversed.

We formally define the modality through the modality membership function

$$\mu_m(a, a_r) \to \{0, 1\}. \tag{2.14}$$

If an article a_r is directly related with a in modality m, the modality membership function returns 1, otherwise it returns 0. For example, for the out-link modality, $\mu_m(a, a_r) = 1$ if and only if there is a hyperlink from a pointing to a_r.

Let us define the modality graph $G_m = (V, E)$, where the set of vertices V is a set of articles in Wikipedia $\mathcal{A}_{wiki}$ and the set of edges E is defined by the modality membership function μ_m:

$$E = \{(a, a_r) : \mu_m(a, a_r) = 1, a \in \mathcal{A}_{wiki}, a_r \in \mathcal{A}_{wiki}\}. \tag{2.15}$$

The graph is crawled starting from the entity article, following in the direction of edges up to a predefined distance – as measured by the number of edges traversed. Each article a_r observed by the crawler is retained along with the path from the entity article taken by the crawler to arrive at this article. The distance of an article (a vertex) from the entity article is not measured by the shortest path, but by the length of the path. If the same article is observed multiple times, it is also retained. We call all the articles for which the length of the path taken by the crawler from the entity article is l as *articles on level* l.

Following this intuition, modality membership function μ_m is used to define the set $\mathcal{A}^a_{m,l}$ which contains walk-article tuples related with the entity article a on level l in modality m. We call this set an *ml*-band for article a.

$$\mathcal{A}^a_{m,l} = \begin{cases} \{\langle \emptyset, a \rangle\} & l = 0, \\ \{\langle W, a_r \rangle | W \text{ is a directed walk in } G_m : W = (a, v_0), \ldots, (v_{l-1}, a_r)\} & l > 0. \end{cases} \tag{2.16}$$

A *directed walk* of length k in a graph G is a sequence of k edges $e_1, \ldots e_k, e_i = (v_t, v_h)$ of G, where the head vertex v_h of the edge e_i is the tail vertex v_t of the edge e_{i+1}. The tail vertex of e_1 is called the *origin* of the walk and the head vertex of edge e_k the *terminus* of the walk.

Note that level l of the *ml*-band $\mathcal{A}^a_{m,l}$ corresponds to the *length* of the walk in all elements contained in the *ml*-band. A special case is for $l = 0$, where $\emptyset$ denotes a *trivial walk* – a walk of length 0.

A practically important consequence of Eq. (2.16) is that one article a_r can be present multiple times in the *ml*-band for a if $l > 1$ and only once if $l = 0$ or if $l = 1$:

- For $l = 0$ this trivially follows from Eq. (2.16).
- For $l = 1$ consider two vertices a and a_r. There is either edge (a, a_r) between these two vertices or no edge and hence there is only one possible walk of length 1 from the entity page a to a_r. As a consequence for a_r there is only one distinct tuple $\langle (a, a_r), a_r \rangle$ in the *ml*-band.
- For $l > 1$ there are multiple distinct walks with origin a and the same terminus a_r. Consider for example $(a, a_1), (a_1, a_r)$ and $(a, a_2), (a_2, a_r)$.

The maximum level for an *ml*-band in modality m is set externally for each modality and is denoted as L^m_{max}.

The set of all articles involved in training for class c can be defined as follows:

$$\hat{\mathcal{A}}_c = \bigcup_{\substack{m \in \mathcal{M}_{train} \\ l = 0 \ldots L^m_{max}}} \mathcal{A}^a_{m,l}(2), \tag{2.17}$$

where $\mathcal{A}^a_{m,l}(2)$ refers to the second component of the tuples in $\mathcal{A}^a_{m,l}$.

The set of all articles involved in training $\hat{\mathcal{A}}_{train}$ can be defined as follows:

$$\hat{\mathcal{A}}_{train} = \bigcup_{c \in C} \hat{\mathcal{A}}_c. \tag{2.18}$$

For an example refer to Subs. 2.3.1.

2.2.3. Aggregation

Let us repeat that the BOA for both unlabeled instance x and target class c (commonly referred to as entity e) are N-dimensional term-weight vectors $\beta(x)$ and $\beta(c)$. The dimension N is determined by function σ.

In order to foster readability, function β is split to "smaller" functions β^{MOD}, β^{INS}, β^{ML} and β^{TWF}.

BOA representation is computed as a weighted average of the term-weight vectors for individual modalities:

$$\beta(e) = \sum_{m \in M} W_m \beta_m^{MOD}(e), \tag{2.19}$$

where modality weight W_m is a parameter set separately for each phase (training/classification) and modality. Modality weights are externally set so that $\sum_m W_m = 1$. Note that in the training phase $M = \mathcal{M}_{train}$ and in the classification phase $M = \mathcal{M}_{test}$.

Function β_m^{MOD} aggregates BOAs constructed for individual *ml*-bands of the modality m:

$$\beta_m^{MOD}(e) = \sum_{l=0}^{L_{max}^m} W_{m,l} \beta_{m,l}^{INS}(e), \tag{2.20}$$

where $W_{m,l}$ is a weight for level l and modality m, the weight is a parameter externally preset so that $\sum_{l=0}^{L_{max}^m} W_{m,l} = 1$.

The function $\beta^{INS}(e)$ aggregates term-weight vectors across entity articles and performs normalization:

$$\beta_{m,l}^{INS}(e) = \sum_{a \in \bar{\mathcal{A}}_e} \frac{1}{|\bar{\mathcal{A}}_e||\beta_{m,l}^{ML}(a)|} \beta_{m,l}^{ML}(a), \tag{2.21}$$

where $\bar{\mathcal{A}}_e$ denotes a set of entity articles given for e as defined in the mapping step. The denominator normalizes the contributions made by the BOA representations of the individual entity articles a by scaling the term-weight vector in level l by the reciprocal of a product of the L1 norm of the BOA vector and the number of entity articles.

If $\bar{\mathcal{A}}_e$ contains only one element, as it is typically when e is an unlabeled instance, Eq. (2.21) can be simplified:

$$\beta_{m,l}^{INS}(e) = \frac{\beta_{m,l}^{ML}(a)}{|\beta_{m,l}^{ML}(a)|} \quad \text{if } |\bar{\mathcal{A}}_e| = 1. \tag{2.22}$$

The function $\beta_{m,l}^{ML}$ aggregates term-weight vectors. Our framework actually supports multiple aggregation functions. Nevertheless, for the sake of clarity, at this point we will make the choice of using the *weighted arithmetic average* to perform the aggregation. A generic aggregation operator is introduced in Subs. 2.3.5 along with additional specific aggregation functions.

$$\beta_{m,l}^{ML}(a) = \sum_{t \in T_m} W_{m,l,t} \beta_{m,l,t}^{TWF}(a), \tag{2.23}$$

where T_m is a *sequence*[5] of term-weighting functions defined for given phase (training/classification) and modality m, the weight $W_{m,l,t}$ is a parameter set for given phase, modality m, level l and term-weighting function τ_t so that $\sum_t W_{m,l,t} = 1$.

Finally, function β^{TWF} aggregates articles within one *ml*-band. This function depends on the type of term-weighting function. For a detailed discussion please refer to Subs. 2.3.4. Here,

[5]While the order is not important for weighted arithmetic average, it is considered by aggregation functions introduced in Subs 2.3.5.

we give the definition for term frequency in place of the term-weighting function:

$$\beta^{TWF}_{m,l,t}(a) = \sum_{\langle W,a_r\rangle \in \mathcal{A}^a_{m,l}} \tau_t(a_r), \tag{2.24}$$

where expression $\langle W, a_r\rangle \in \mathcal{A}^a_{m,l}$ denotes that we are selecting walk-article tuples included within the ml-band defined by the modality m and level l, and τ_t is a term-weighting function identified by the index t. The term-weighting function $\tau_t \in T_m$ is used to represent an article as a vector of term weights.

$$\tau_t(a) \to R^N. \tag{2.25}$$

The term-weighting functions used in place of τ_t include e.g. TF, IDF or new term-weighting functions derived from WordNet (refer to Subs 2.3.3). The component on the i-th position in $\tau_t(a)$ corresponds to a weight associated with term on the i-th position as given by $\sigma(\hat{\mathcal{A}}_{train})$ in Eq. (2.3).

2.3. Detailed Description

In this section, selected topics relating to the BOA representation are further elaborated. Modality membership function μ_m is described in more detail in Subs. 2.3.1, term selection function σ is described in Subs. 2.3.2, various term weighting function are described in Subs. 2.3.3, bulk operations on term-weight vectors with matrices in Subs. 2.3.4, term weight aggregation functions in Subs. 2.3.5 and the classification step in Subs. 2.3.6.

2.3.1. Modality Membership μ

Modality membership function $\mu(a, a_r) \to \{0, 1\}$ expresses if article a_r is considered related to a ($\mu = 1$) or not ($\mu = 0$). Several modality membership functions are suggested below. Article a is evaluated as related ($\mu(a, a_r) = 1$) to a_r, $a \neq a_r$:

- $\mu_{out\text{-}link}(a, a_r) = 1$ iff a links to a_r,
- $\mu_{in\text{-}link}(a, a_r) = 1$ iff a_r links to a,
- $\mu_{same\ category}(a, a_r) = 1$ iff a and a_r share the same category,
- $\mu_{similar\ content}(a, a_r) = 1$ iff the textual similarity of a and a_r exceeds a preset threshold.

The page is considered as unrelated to itself:

$$\mu(a, a_r) = \mu(a_r, a) = 0 \text{ if } a = a_r. \tag{2.26}$$

Note that except this case, $\mu(a, a_r) = \mu(a_r, a)$ does not generally hold.

Composite membership function definitions are also easily imaginable. For example, modality $\mu_{shared\ category\ out\text{-}link}$ can be expressed as a union of constraints imposed by $\mu_{same\ category}$ and $\mu_{out\text{-}link}$. This applies e.g. to:

- $\mu_{firstpara\ out\text{-}link}(a, a_r) = 1$ iff a links to a_r and the link from a to a_r is contained in the first paragraph of a,

- $\mu_{related\ out\text{-}link}(a, a_r) = 1$ iff a links to a_r and there is an article a_c linking to a and a_r, $a_r \neq a \neq a_c$,
- $\mu_{backlinking\ out\text{-}link}(a, a_r) = 1$ iff a links to a_r, a_r links to a,
- $\mu_{shared\ category\ out\text{-}link}(a, a_r) = 1$ iff a links to a_r and a and a_r share the same category.

Various article selection methods corresponding to some of the modality membership functions drafted above have in fact been suggested in the literature. For example, a notion very closely related to $\mu_{backlinking\ out\text{-}link}$ and $\mu_{firstpara\ out\text{-}link}$ was suggested in [Cuc07] and $\mu_{related\ out\text{-}link}$ is used in the Lucene Search Mediawiki Extension (refer to Sec. 2.6).

The BOA implementation introduced later in this chapter features $\mu_{out\text{-}link}$, $\mu_{in\text{-}link}$ and $\mu_{same\ category}$. It also supports $\mu_{similar\ content}$, but it can be often used only in conjunction with any of the former functions to create a composite membership function.

The following example shows how is a modality membership function used to define the sets $\mathcal{A}^a_{m,l}$ for $l = 0, 1, 2$.

Example 2.2 (Modality membership function and sets $\mathcal{A}^a_{m,l}$). Consider a crawling task in modality m up to level $L^m_{max} = 2$. Assume that the Wikipedia consists of five articles $\{a_1, a_2, a_3, a_4, a_5\}$.

Let there be article a_1 such that there are exactly two different articles a_2 and a_3 related to it:

$$\mu_m(a_1, a_2) = 1$$
$$\mu_m(a_1, a_3) = 1$$

It follows from Eq. (2.16):

$$\mathcal{A}^{a_1}_{m,0} = \{\langle \emptyset, a_1 \rangle\}$$
$$\mathcal{A}^{a_1}_{m,1} = \{\langle (a_1, a_2), a_2 \rangle, \langle (a_1, a_3), a_3 \rangle\}.$$

Let a_4 be the only article related to a_2 and the only article related to a_3. In that case, it follows from Eq. (2.16):

$$\mathcal{A}^{a_1}_{m,2} = \{\langle (a_1, a_2), (a_2, a_4), a_4 \rangle, \langle (a_1, a_3), (a_3, a_4), a_4 \rangle\}$$

Although a_4 is included twice on level 2, the set $\hat{\mathcal{A}}_{train}$ contains according to Eq. (2.18) only the unique articles:

$$\hat{\mathcal{A}}_{train} = \{a_1, a_2, a_3, a_4\}$$

2.3.2. Term Selection σ

Term selection can be performed either globally by removing some words from articles before subsequent term vector operations, or locally for each bag of articles. In the latter case, the term is pruned by putting 0 to the position corresponding to the term.

$$\sigma(\mathcal{A}_{train}) \rightarrow \langle term_1 \dots term_N \rangle \tag{2.27}$$

Technically, it is advantageous to precede term weighting with term selection (term pruning) in order to avoid unnecessary operations. The most widely used methods for term selection include:

- negative list of words (usually called stop-word list) – words on the list are not selected,
- positive list of words – only words on the list are selected,
- frequency-based pruning – words either above or below preset frequency characteristic are discarded.

Application of the first two options in the BOA setting is straightforward. Concerning frequency-based pruning, we propose a slight modification detailed below. In this subsection we also include some details on lemmatization, as a related dimensionality reduction technique.

Frequency-based Term Pruning

Document frequency expresses the number of documents in which the term occurs. According to [FS06] (p. 69): "Experimental evidence suggests that using only the top ten percent of the most frequent words does not reduce the performance of classifiers." This quote refers document frequency (DocFreq).

If the number of distinct terms found in all training data exceeds a preset threshold, terms are sorted according to term frequency and the most frequently occurring terms are kept.

Lemmatization

This feature extraction technique replaces the terms with their lemmas. A lemma is a dictionary form of the word. As a part of the lemmatization process only some parts of speech (most commonly nouns) can be retained. Lemmatization has similar effects as the above mentioned term selection techniques in that it also reduces the number of terms.

In [BKV03] it is observed that lemmatization does not seem to improve the overall accuracy on English and Spanish corpora, the increase in recall is offset by a significant drop in precision. In contrast, [ZAAS05] notes that lemmatization on a Basque corpus results in a marked improvement of performance. This evidence suggests that the impact of lemmatization in text categorization tasks seems to be linked to the morphological richness of the particular language.

2.3.3. Term Weighting τ

The weight function $\tau(a) \rightarrow R^N$ represents the article a as a vector of term weights.

The elementary term-weighting functions considered are:

- τ_{tf} term frequency,
- τ_{idf} inverse document frequency,
- τ_{wnet} WordNet similarity with the target concept.

Term Frequency

Term Frequency (TF) is the number of times that the given term appears in the document.

A variant of the term frequency may consider position of the word in the document. The first paragraph and first sentence in particular receive special attention in the Wikpedia manual of style (more discussion in Subs. 5.1) and as a consequence it is reasonable to expect that it will contain words closely related to the entity described in the article. Giving boost to the first paragraph, sometimes called a "gloss" [SP06], is a straightforward extension. The first paragraph of a Wikipedia article contains usually the definition of the article subject, it can be therefore expected to contain more relevant words than the rest of the text. Special treatment of first sentence can be found e.g. in [Cuc07] and [KT07] and of the gloss in [SP06].

Inverse Document Frequency

Inverse Document Frequency (IDF) of term t is computed with the following formula:

$$IDF(t) = 1 + log\left(\frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}| + 1}\right) \tag{2.28}$$

In the context of the BOA computation, the set of documents $\mathcal{D}$ on which the computation is performed may either refer to *all articles in Wikipedia*, or only to all *training* articles. An article needs to appear in at least one BOA of a labeled instance C to be considered as a training article. The article is counted only once even if it appears in multiple BOAs. We consider three variations of the IDF weight:

IDF_{ALL} is computed over entire Wikipedia:

$$IDF_{ALL}(t) = 1 + log\left(\frac{|\mathcal{A}_{wiki}|}{|\{a \in \mathcal{A}_{wiki} : t \in a\}| + 1}\right), \tag{2.29}$$

where $t \in a$ is a shorthand for expressing that term t appears in article a.

IDF_{TRAIN} is computed only over articles involved in training:

$$IDF_{TRAIN}(t) = 1 + log\left(\frac{|\hat{\mathcal{A}}_{train}|}{|\{a \in \hat{\mathcal{A}}_{train} : t \in a\}| + 1}\right). \tag{2.30}$$

IDF_{BOA} is computed only over bag of articles of classes involved in training. All articles in one BOA are considered as one document. The number of documents $|\mathcal{D}|$ for the purpose of Eq. (2.28) is equal to the number of training classes, and the denominator is increased by 1 with every target class that has a BOA representation with a document containing term t:

$$IDF_{BOA}(t) = 1 + log\left(\frac{|C|}{|\{c \in C : (\exists a \in \hat{\mathcal{A}}_c : t \in a)\}| + 1}\right). \tag{2.31}$$

Generally, $|\mathcal{A}_{wiki}| >> |\hat{\mathcal{A}}_{train}| >> |C|$.

It is not clear when to choose which weight. For datasets that are narrow in focus and tend to contain specialized vocabulary IDF_{TRAIN} may be more discriminatory. Consider the following case. In the Czech traveler dataset (refer to Subs. 6.1.3), words like "sea" or

	Wikipedia document frequency	IDF_{ALL}
Albanian	84 84	2.81
monastery	25 281	2.33
dragon	43 534	2.10
sea	157 502	1.54

Table 2.1.: IDF illustrative example – several entities from the Czech Traveler dataset.

"dragon"[6] are very descriptive as indicated by the fact that each appears only in one image title. These words frequently appear in other contexts. In contrast, words like "Albanian" or "monastery" occur quite frequently in this dataset, but are quite infrequent in Wikipedia.

Table 2.1 shows that the IDF_{ALL} weights for "Albanian" or "monastery" are much higher than of "sea" or "dragon", which is contrary to the intuitive distribution of these words in our dataset.

Applying IDF_{ALL} may be advantageous in the case when there is a risk of picking up a wrong sense for the unlabeled entities. For example, if the entity "kvas sale" is erroneously mapped to Norwegian village "Kvås". In the "Kvås" article, many words (e.g. Norway, Lyngdalen) would be unique to the article and not present in any other training entity's BOA representation and hence get high IDF_{TRAIN} and IDF_{BOA} weights. On the other hand, the IDF_{ALL} weights would not be boosted by the wrong sense selection. Interestingly, high IDF_{TRAIN} and IDF_{BOA} values for many terms in the document are indicative of wrong disambiguation.

IDF_{TRAIN} and IDF_{BOA} can be more effective when the risk of choosing a wrong sense is small, but the individual target labels are close to one another and IDF_{ALL} does not provide enough discriminatory power. This can be seen in the example given in Table 2.1.

To conclude the discussion, IDF_{ALL} is more universal and could thus be more suitable for disambiguation purposes (refer to Sec. 2.4) while IDF_{TRAIN} and IDF_{BOA} could be more apt for classification of a disambiguated entity.

WordNet Weights

This subsection deals with using WordNet for term weighting. WordNet has been applied for assessing term significance and term weighting before. For example in [SKA09], the following types of information are used to determine the generality or specificity of a term: number of senses, number of synonyms, level number and number of children. In term weighting, we use also many pieces of this information, but not explicitly – they are embedded in a WordNet similarity measure. In fact, we use WordNet in much the same way as in SCM (refer to Sec. 1.2.3).

The difference is that in SCM this WordNet similarity is the final outcome, while in BOA it is used as a weight in the similarity computation. Since many entity-term similarities contribute to the overall class-instance similarity, the hypothesis is that the occasional misclassifications and inconsistencies should be absorbed in favour of the prevailing correct sense.

[6] Appearing in place name "Jetee du Dragon"

In BOA similarity is computed between the term and a WordNet synset, which is optionally associated with the target class (refer to Subs. 2.1.2). Consequently, each target class is associated with a different WordNet term-weight vector (provided that classes are mapped to different WordNet synsets).

Example 2.3 (WordNet similarity as term-weighting function). Consider a setting with two target classes. The classes are represented according to Eq. (2.1):

$$c = \langle \text{class name}, \mathcal{A}_c, \mathcal{NP}_c(, \text{WordNet synset}) \rangle.$$

Using this as a "template", we get:

$$c_1 = \langle \text{footballer}, \{\text{footballer}\}, \emptyset, \text{football\_player\#1} \rangle$$
$$c_2 = \langle \text{basketball player}, \{\text{basketball player}\}, \emptyset, \text{basketball\_player\#1} \rangle.$$

Assuming that that there are only three distinct words in all articles involved in training:

$$\sigma(\mathcal{A}_{train}) = \langle \text{'football', 'basketball', 'Maradona'} \rangle.$$

The distribution of words between the target classes is as follows. The bag of articles for football player contains only two words ('football' and 'Maradona') and the bag of articles for basketball player contains again only two words ('football' and 'basketball').

We get the following term-weight vector for $\tau_{wnet}(c_1)$:

$$\langle wsim(\text{football\_player\#1, football}), 0, wsim(\text{football\_player\#1, Maradona}) \rangle.$$

We get the following term-weight vector for $\tau_{wnet}(c_2)$:

$$\langle wsim(\text{basketball\_player\#1, football}), wsim(\text{basketball\_player\#1, basketball}), 0 \rangle.$$

Here, $wsim$ refers to a WordNet similarity measure, e.g. Lin (refer to 4.2.4). The zeros in these term-weight vector correspond to words that are not present in any of the articles involved in the bag for the given class.

If a word is not in WordNet ("Maradona" in the example) the similarity is 0. Such words can be removed in a global pruning step.

2.3.4. Computing Term Weight Vectors with Matrices

In this subsection, we will show how can be β^{TWF} computed using matrix manipulation. The function $\beta^{TWF}_{m,l,t}(a) \rightarrow R^N$ serves for aggregation of articles on a given level represented with the term-weighting function τ_t.

Typology of term-weighting functions

We distinguish three types of term-weighting functions: *global*, *class* and *article*.

Global term-weighting function associates one value with a term irrespective of its scope (*ml*-band, modality, phase). All types of inverse document frequency introduced in Subs. 2.3.3 are of this type.

Class term-weighting function associates one value with a term per target class. All types of WordNet term-weighting functions (refer to Subs. 2.3.3) are of this type.

Article term-weighting function associates one value with a term per article. Term frequency is of this type.

Unlike the *article* type, both global and class types do not directly depend on term frequencies in the *ml*-band. The *global*-type term weights are computed once and then reused in all *ml*-bands, the *class* type term weights are computed once per target class and then reused in all *ml*-bands related to that class, the *article* type term weights are computed for each article and then reused in all *ml*-bands involving this article.

Term Weights

Below, we will define the term-weight matrix D for each term-weighting function type. This matrix is later used to compute β_{TWF}.

For the remainder of this subsection, let us denote the number of articles in Wikipedia $M = |\mathcal{A}_{wiki}|$ and the dimensionality of the term-weight vector $N = |\sigma(\hat{\mathcal{A}}_{train})|$.

Global Term-weighting Function

$$D = \begin{bmatrix} d_1 & d_2 & \dots & d_N \end{bmatrix} \tag{2.32}$$

Class Term-weighting Function

$$D = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1N} \\ d_{21} & d_{22} & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{|C|1} & d_{|C|2} & \dots & d_{|C|N} \end{bmatrix} \tag{2.33}$$

where D_{ij} is a weight of $j-th$ term for class i and $|C|$ is the number of training classes.

Although *class*-type term-weighting functions can also be used in the classification phase, it is advisable to use them only in the training phase for performance reasons. WordNet measures are relatively expensive to compute and if they are not used in the classification phase, the values for terms which do not occur in any of the articles in the bag of the target class need not be computed.

Article Term-weighting Function

$$D = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1N} \\ d_{21} & d_{22} & \dots & d_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M1} & d_{M2} & \dots & d_{MN} \end{bmatrix} \tag{2.34}$$

where D_{ij} is a weight of $j-th$ term in article i. This matrix is commonly called *document-term* matrix in information retrieval. Here, M is the total number of documents (articles) in A_{wiki}.

Modalities

A modality m can be expressed as an incidence matrix over Wikipedia articles.

$$ {}^{m}B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1M} \\ b_{21} & b_{22} & \dots & b_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \dots & b_{MM} \end{bmatrix} \tag{2.35} $$

where $b_{i,j} = \mu_m(a_i, a_j)$ and a_i, a_j refers to article identified by index i and j respectively.

It follows from Eq. (2.26) that the ${}^{m}B$ matrix will have zeros on the main diagonal.

Matrix Multiplication

The computation depends on the type of the term-weighting function:

$$ \beta^{TWF}_{m,l,t}(a_i) = \begin{cases} ({}^{m}B^{l}D)_{(i*)} & \text{if } t \text{ is of article type,} \\ D_{(c*)} & \text{if } t \text{ is of class type and } c \text{ is the class in scope,} \\ D & \text{if } t \text{ is of global type,} \end{cases} \tag{2.36} $$

where ${}^{m}B^{l}$ is the incidence matrix for the modality m raised to l. Note that ${}^{m}B^{0} = I$, where I is an identity matrix.

This computation relies on the following relation of powers of incidence matrices and walks in the graph: If n is a positive integer, G^n gives the number of paths in graph G of exactly n arcs from each vertex to each other vertex [Tuc89].

From this definition it follows that there is a close correspondence between i-th row in ${}^{m}B^{l}$ and the ml-band: the value of $({}^{m}B^{l})_{ij}$ expresses the number of times article a_j appears in ml-band for article a_i. This can be formally expressed as:

$$ ({}^{m}B^{l})_{ij} = |\{\langle W, a_j\rangle : \langle W, a_j\rangle \in \mathcal{A}^{a_i}_{m,l}\}|. \tag{2.37} $$

2.3.5. Term-Weight Representation of *ml*-band β^{ML}

The term-weight aggregation function β^{ML} allows to create composite term-weighting functions by aggregating individual term-weighting functions. An example of such a measure is TF-IDF, which is created by aggregating TF and IDF using multiplication as the aggregating function. The result of this function is a term-weight vector representing the ml-band

There are multiple ways this aggregation can be done. The input for aggregation are multiple term-weight vectors of length N, technically there can also be only one such vector.

We define a generic aggregation function:

$$ \beta^{ML}_{m,l}(a) = aggreg(\, \beta^{TWF}_{m,l,1}(a), \dots, \beta^{TWF}_{m,l,i}(a), \dots, \beta^{TWF}_{m,l,|T_m|}(a), W_{m,l,1}, \dots, W_{m,l,|T_m|}\,). \tag{2.38} $$

To simplify the discourse, let us denote the $n = |T_m|$ term-weight vectors as $\vec{x}_1, \dots, \vec{x}_i, \dots, \vec{x}_n$, where $\vec{x}_i = \langle x_{i,1}, \dots, x_{i,N}\rangle$. Similarly, the associated weights $W_{m,l,1}, \dots, W_{m,l,|T_m|}$ will be in this subsection addressed as $\vec{w} = \langle w_1, \dots, w_n\rangle$. Eq. (2.38) can thus be restated as:

$$ \bar{x} = aggreg(\vec{x}_1, \dots, \vec{x}_i, \dots, \vec{x}_n, \vec{w}) \to R^N. \tag{2.39} $$

The weight subvector $\vec{w} = \langle w_1, \ldots, w_n \rangle$ is normalized:

$$|\vec{w}| = \sum_{j=1}^{n} w_j = 1. \tag{2.40}$$

In our framework the aggregation is done on *per-component* basis:

$$\bar{x_i} = aggreg(x_{1,i}, \ldots, x_{j,i}, \ldots, x_{n,i}, \vec{w}) \to R, \tag{2.41}$$

where $\bar{x_i}$ is the i-th component (term-weight) of the aggregation result $\bar{x}$ and $x_{j,i}$ is the i-th component of j-th term-weight vector.

Important notation note The remainder of this subsection will deal with operations pertaining to only one component of the term-weight vectors x and $\bar{x}$, we will therefore omit the index i: $x_{j,i}$ becomes x_j and $\vec{x_i}$ becomes $\vec{x}$.

Arithmetic Average

Weighted arithmetic average in place of the β^{ML} was shown in Eq. (2.23). Using the simplified notation introduced in this subsection, it can be restated as:

$$\bar{x} = \sum_{j=1}^{n} w_j x_j. \tag{2.42}$$

The denominator can be omitted because according to Eq. (2.40) the weights are normalized.

Although arithmetic average is intuitive and simple to compute, it is not suitable when aggregating values from several heterogeneous sources. Since term-weighting functions are quite disparate ranging from term frequency with integer value range to WordNet similarity measures which are float numbers either in $\langle 0; 1 \rangle$ or $\langle 0; \text{inf} \rangle$ range, arithmetic average is not suitable. Normalizing the values will not solve this problem [FW86].

Weighted Geometric average

Geometric average is strongly suggested in [FW86] for averaging normalized values. Weighted average mean for component i is defined as:

$$\bar{x} = \left(\prod_{j=1}^{n} x_j^{w_j} \right)^{1/\sum_{j=1}^{n} w_j}. \tag{2.43}$$

According to Eq. (2.40) this can be simplified:

$$\bar{x} = \prod_{j=1}^{n} x_j^{w_j}. \tag{2.44}$$

A certain disadvantage of using geometric average is that if one of the aggregated values is zero, then the result is also zero irrespective of the weight of the aggregated value.

Example 2.4 (Cons of geometric average). Consider aggregating term frequency with a WordNet term weight. Let the term frequency value be $x_1 = 3$ and the value for WordNet similarity-based term weight $x_2 = 0$. Assuming that values of WordNet similarity measures are fallible, the value 0 may not necessarily mean that the real value is 0. Even if we project the uncertainties about the reliability of the WordNet term weight into our weighting scheme, with term frequency being assigned the weight $w_1 = 0.9$, and WordNet being assigned weight $w_2 = 0.1$, the result of aggregation of the example two term weights with geometric average will still be zero:

$$\bar{x} = x_1^{w_1} \times x_2^{w_2} = 3^{0.9} \times 0^{0.1} = 0 \tag{2.45}$$

This result underpins the unsuitability of geometric average in volatile settings, where one of the aggregated values can be expected to be (erroneously) zero.

In addition to this, it is also noted in [FW86] that any measure of the mean value of data is misleading when there is larger variance. The same source recommends to use also the minimum and maximum of the data in aggregation operators.

Custom Aggregator Operators

In an attempt to address some of the shortcomings of geometric and arithmetic means[7], we propose two custom aggregator operators. Both operators give the first aggregated value x_1 a special treatment: it is not (directly) weighted. The remaining values are added to it in a weighted manner in a recurrent fashion. The key difference between the two aggregators is whether the result can drop below the first value under the influence of the remaining values.

For both aggregators, the result of aggregation of all the n values $x_1, \ldots, x_n$ is the value of $\bar{x}_1^n$, where the value 1 indicates the index of the first aggregated value and n the index of the last aggregated value.

Also note that both proposed aggregator functions ignore weight w_1, which is associated with the first aggregated value x_1. Since our generic aggregate function assumes the existence of w_1, w_1 should be set to 0 not to disturb weight normalization.

Custom Aggregator 1

$$\bar{x}_1^j = \begin{cases} (1 - w_j)\bar{x}_1^{j-1} + w_j x_j \bar{x}_1^{j-1} & j > 1, \\ x_1 & j = 1. \end{cases} \tag{2.46}$$

This aggregator does allow the result to drop below the first weight. For properties refer to Table 2.2.

Custom Aggregator 2

$$\bar{x}_1^j = \begin{cases} \bar{x}_1^{j-1} + w_j x_j \bar{x}_1^{j-1} & j > 1, \\ x_1 & j = 1. \end{cases} \tag{2.47}$$

[7] Harmonic mean was not considered, because it requires *positive* real numbers.

j	>1				1
w_j	0	1	$(0;1)$		
x_j		0	>1	$(0;1)$	
$\bar{x}_1^j$	$\bar{x}_1^{j-1}$	0	$\geq \bar{x}_1^{j-1}$	$(0;\bar{x}_1^{j-1})$	x_1

Table 2.2.: Custom Aggregator 1 – the effect of the value of j, w_j, x_j on $\bar{x}_1^j$

j	>1				1
w_j	0	1	$(0;1)$		
x_j		0	>1	$(0;1)$	
$\bar{x}_1^j$	$\bar{x}_1^{j-1}$	$\bar{x}_1^{j-1}$	$\geq \bar{x}_1^{j-1}$	$(0;\bar{x}_1^{j-1})$	x_1

Table 2.3.: Custom Aggregator 2 – the effect of the value of j, w_j, x_j on $\bar{x}_1^j$

This aggregator behaves similarly as the previous aggregator, but it does not allow the result to drop below the first weight:

$$w_j = 1 \wedge x_j = 0 \implies \bar{x}_1^j = \bar{x}_1^{j-1} \tag{2.48}$$

Properties of the Custom Aggregator 2 are concisely depicted on Table 2.3.

2.3.6. Classification

The gist of the BOA approach is in the representation.

Both the unlabeled instance x and the target classes $c \in C$ are represented with the bag-of-articles function $\beta \to N$:

$$class(x) = \arg\max_{c \in C} sim(\beta(c), \beta(x)). \tag{2.49}$$

where sim refers to selected similarity measure.

As the BOA similarity measure sim, the implementation supports cosine similarity and dot product.

Cosine similarity

$$sim(x, y) = \frac{\sum_{j=1}^{N} x_j\, y_j}{\sqrt{\sum_{j=1}^{N} (x_j)^2}\, \sqrt{\sum_{j=1}^{N} (y_j)^2}}, \tag{2.50}$$

where x_j is a j-th component of $\beta(x)$, y_j is a j-th component of $\beta(y)$ and N is the length of the BOA vector. This equation is applied to obtain similarity of BOA representations of an unlabeled instance and each of the target classes.

The dimensionality of the BOA vectors N may be very high. If there is a high number of articles involved in the training, then it can be in our experience expected that $10^5 < N < 10^6$. One measure to address this problem is term selection introduced in Subs. 2.3.2.

Another optimization is aimed at reducing the number of operations when computing the cosine similarity measure. In order to avoid repeated computation of the denominator in Eq. (2.50), it is advantageous to perform *cosine normalization*[8] of the BOA representation by dividing each term weight w by a factor representing Euclidean vector length (L2 norm of the vector):

$$\beta_2(e) = \frac{\beta(e)}{|\beta(e)|_2}, \tag{2.51}$$

where symbol $|\beta(e)|_2$ denotes L2 norm of the vector $\beta(e)$:

$$|\beta(e)|_2 = \sqrt{\sum_{i=1}^{N} x_i^2}, \tag{2.52}$$

where x_i is an i-th component of $\beta(e)$.

The formula 2.50 can be then simplified to a dot product of the L2-normed BOA representations of the target class and the unlabeled instance:

$$sim(c, x) = \sum_{i=1}^{n} c_i \, x_i = \beta_2(c)(\beta_2(x))^T. \tag{2.53}$$

Dot Product Note that the input $\beta(e)$ is L1-normalized:

$$\beta(e) = |\beta(e)|_1 = \sum |x_i| = \sum x_i = 1, \tag{2.54}$$

which follows from the fact that $\beta^{INS}(e)$ is L1-normalized and the subsequent operations in Eq. (2.20) preserve this. While this property is lost by the L2-normalization to $\beta_2(e)$ in cosine similarity computation, it can be exploited if the input vectors are compared using dot-product:

$$sim(c, x) = \sum_{i=1}^{n} c_i \, x_i = \beta(c)(\beta(x))^T. \tag{2.55}$$

Our experiments on WordSim353 dataset showed that dot product produces consistently better results than cosine similarity.

2.4. Disambiguation

This section describes the use of BOA representation for mapping unlabeled entities onto Wikipedia articles. The baseline disambiguation function δ_{mfs} takes the most-frequent sense (MFS) approach, selecting the article with the highest rank assigned by the ranking function to each unlabeled entity; each entity being ranked independently. While the motivational problem for the BOA entity classifier stated in the introduction asserts that the left and right context for an entity is not available, it can be in many classification scenarios assumed that the unlabeled entities come from the same dataset and hence share the same global context.

[8] This operation is called *cosine transformation* in [Fri10], *cosine normalization* in [GLM05, SB88].

Example 2.5 (Disambiguation on Israeli images). Consider noun phrases extracted from captions of the Israeli images dataset (refer to [BJ07] and Subs. 6.1.3). This dataset contains 1823 image-captions mostly thematically related to Israel. After the analysis of the captions with an entity extraction system, the result includes among others the following noun phrases: np_1 : *The Church Of St Joseph*, np_2: *Jerusalem* and np_3: *Wadi Qelt*. For the first entity np_1, the first sense $s_{1,1} = \delta_{mfs}(np_1)$ is a Wikipedia article *St. Joseph's Catholic Church (Lacona, Iowa)*, which is obviously incorrect. The correct sense is probably $s_{1,14}$ =*St. Joseph's Church, Nazareth*. With noun phrases np_2 (assigned sense vector $\langle s_{2,1}, s_{2,2}, \ldots\rangle$) and np_3 ($\langle s_{3,1}, s_{3,2}, \ldots\rangle$), the most frequent sense assumption works well and $s_{2,1}$ and $s_{3,1}$ are correct.

Algorithm 6 BOA Heuristic Disambiguation Algorithm

Input: $S := \{s_1, \ldots, s_u\}$, where $s_i = \langle s_{1,1}, \ldots, s_{1,n}\rangle$,
$MAXIT$, $MAXSIM$ // possible senses of entity 1...u
Output: $\langle s_{1,i}, \ldots, s_{u,j}\rangle$ //disambiguated titles
for all s_i in S **do**
append $s_{i,1}$ to DIS
end for
$CLUS$ = cluster(S)
while $it < MAXIT$ and DIS not empty **do**
//Identify the biggest outlier
$i, j := arg\ min\ _{i,j}\ sim(s_{i,j}, c_i); c_i \in CLUS, s_{i,j} \in DIS$
// finish if it is not far enough
if $sim(s_{i,j}, c_i) > MAXSIM$ **then**
break
end if
// Try to get a closer sense
$k := arg\ max_k\ sim(s_{i,k}, c_i), c_i \in CLUS, s_{i,k} \in s_i \in S$
// move to resolved if none found
if $j == k$ **then**
remove $s_{i,j}$ from DIS
put $s_{i,j}$ to RES
// move to resolved if there is a cycle
else if "$s_{i,j}$ => $s_{i,k}$" in $SWAPLOG$ **then**
remove $s_{i,j}$ from DIS
put $s_{i,j}$ to RES
// otherwise swap with a closer sense
else
replace $s_{i,j}$ with $s_{i,k}$ in DIS
put "$s_{i,j}$ => $s_{i,k}$" to $SWAPLOG$
update $CLUS$
end if
$it = it + 1$
end while
return $DIS \cup RES$

The idea for the proposed heuristic disambiguation algorithm is to replace the outlying senses with a sense that better fits with other selected senses. The algorithm contains a clustering step to reflect the fact that the dataset may contain several distinct types of entities, some of which may not share a common context. For example, in the Israeli dataset there are several such entities such as *Rain Pool* or *Apple*. Trying to adjust the sense of these outlying entities with the rest of the collection could result in misclassification.

2.4.1. Disambiguation Algorithm

Alg. 6 presents the pseudocode for the disambiguation algorithm. In each iteration, the algorithm identifies the entity s_i that is farthest from any of the context clusters and tries to find a sense $s_{i,k}$ that would better match any of the context clusters than the currently assigned sense $s_{i,j}$. If no better matching sense is found, the entity is removed from the *DIS* (to DISambiguate) list and put into the *RES* (RESolved) list. If a better matching sense is found, it is first checked if the swap $s_{i,j} \rightarrow s_{i,k}$ has not already occurred in a previous iteration. If this is the case, the entity is removed from *DIS* and put into the *RES* list, otherwise the sense $s_{i,k}$ replaces $s_{i,j}$ in DIS and this operation is recorded into the $SWAPLOG$, a change log of swaps in senses. The algorithm stops either when the DIS list is empty, or when the similarity between the most remote entity and its closest cluster exceeds the $MAXSIM$ parameter, or when the maximum number of iterations $MAXIT$ is reached. After the disambiguation is performed, the classification is done in the way described in the previous section.

The results obtained by this algorithm may be well-below the most frequent sense baseline, depending on the dataset. It is a widely acknowledged fact that it is difficult even for a supervised system to meet this baseline. To illustrate this, [PDKM09] states that

> "unsupervised systems were found to never outperform the most frequent sense (MFS) baseline (a sense assignment made on the basis of the most frequent sense in an annotated corpus), while supervised systems occasionally perform better than the MFS baseline, though rarely by more than 5%".

2.5. Example

This section introduces a toy example, which will demonstrate the BOA representation and BOA classification.

2.5.1. Source Data

We will use a fictitious Wikipedia containing only 6 articles:

$$\mathcal{A}_{wiki} = \{a_1, a_2, a_3, a_4, a_5, a_6\}. \tag{2.56}$$

Within the example only two modalities are considered – in-link and out-link. The link graph corresponding to the out-link modality is depicted in Fig. 2.1 and the corresponding incidence matrix is depicted in Table 2.4. The graph for the in-link modality can be obtained by reversing the direction of the edges or transposing the incidence matrix.

In our toy example, there are only 7 distinct terms. The term-weight matrix for term frequency is depicted in Table 2.5.

	a_1	a_2	a_3	a_4	a_5	a_6
a_1	0	1	1	0	0	0
a_2	0	0	0	1	0	0
a_3	0	0	0	1	0	0
a_4	1	1	0	0	1	0
a_5	0	0	0	1	0	1
a_6	0	0	0	0	0	0

Table 2.4.: BOA Example – Out modality incidence matrix ${}^{out}B$

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
a_1	1	2	0	0	0	0	0
a_2	0	1	2	0	0	0	0
a_3	1	0	0	0	1	0	0
a_4	1	4	0	5	0	0	0
a_5	0	1	5	0	1	1	0
a_6	0	1	5	0	1	1	1

Table 2.5.: BOA Example – Term-weight Matrix for term frequency

2.5.2. Training Phase

There are two target classes:

$$C = \{c_1, c_2\}. \tag{2.57}$$

Let us recall the representation of target class introduced in (2.1):

$$c = \langle \text{class name}, \mathcal{A}_c, \mathcal{NP}_c(\text{,WordNet synset}) \rangle. \tag{2.58}$$

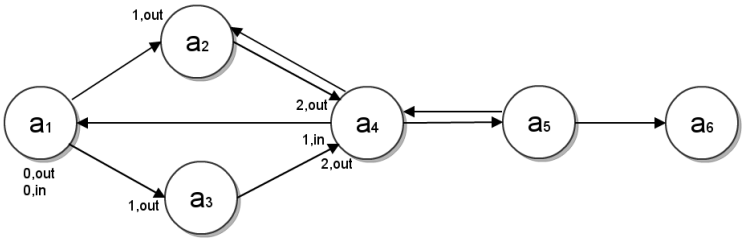

Figure 2.1.: Out-link and in-link modalities for a_1 with marked levels. The labels at link ends are used to denote the level and modality to which the adjacent article node through the link belongs.

In this example, we will not use any WordNet term-weighting function, therefore the optional WordNet synset part is left out:

$$c_1 = \langle \text{class } 1, \{a_1\}, \emptyset \rangle \tag{2.59}$$
$$c_2 = \langle \text{class } 2, \{a_3\}, \emptyset \rangle. \tag{2.60}$$

Each of the target classes is directly mapped to one Wikipedia article. Since no noun phrases were given, the sets NP_{c_1} and NP_{c_2} are empty as denoted by $\emptyset$ in Eq. (2.59) and (2.60).

The set of all entity articles involved in training:

$$\mathcal{A}_{train} = \{a_1, a_3\}. \tag{2.61}$$

Mapping

Since the sets NP_{c_1}, NP_{c_2} are empty, the target classes c_1, c_2 are trivially mapped according to Eq. (2.9) to articles in A_{c_1} and A_{c_2} respectively:

$$\bar{\mathcal{A}}_{c_1} = \mathcal{A}_{c_1} = \{a_1\} \tag{2.62}$$
$$\bar{\mathcal{A}}_{c_2} = \mathcal{A}_{c_2} = \{a_3\}. \tag{2.63}$$

Crawling

The configuration for crawling is to use the *out-link* and *in-link* modalities with maximum levels for out-link modality

$$L^{out}_{max} = 2, \tag{2.64}$$

and for in-link modality:

$$L^{in}_{max} = 1. \tag{2.65}$$

For details refer to Subs. 2.5.2. The articles that appear within the modalities on individual levels are depicted on Fig. 2.1 and also listed according to Eq. (2.16) by *ml*-band below:

$$\mathcal{A}^{a_1}_{out,0} = \{\langle \emptyset, a_1 \rangle\} \tag{2.66}$$
$$\mathcal{A}^{a_1}_{out,1} = \{\langle (a_1, a_2), a_2 \rangle, \langle (a_1, a_3), a_3 \rangle\} \tag{2.67}$$
$$\mathcal{A}^{a_1}_{out,2} = \{\langle (a_1, a_2), (a_2, a_4), a_4 \rangle, \langle (a_1, a_3), (a_3, a_4), a_4 \rangle\} \tag{2.68}$$
$$\mathcal{A}^{a_1}_{in,0} = \{\langle \emptyset, a_1 \rangle\} \tag{2.69}$$
$$\mathcal{A}^{a_1}_{in,1} = \{\langle (a_1, a_4), a_4 \rangle\} \tag{2.70}$$
$$\mathcal{A}^{a_3}_{out,0} = \{\langle \emptyset, a_3 \rangle\} \tag{2.71}$$
$$\mathcal{A}^{a_3}_{out,1} = \{\langle (a_3, a_4), a_4 \rangle\} \tag{2.72}$$
$$\mathcal{A}^{a_3}_{out,2} = \{\langle (a_3, a_4), (a_4, a_5), a_5 \rangle, \langle (a_3, a_4), (a_4, a_2), a_2 \rangle\} \tag{2.73}$$
$$\mathcal{A}^{a_3}_{in,0} = \{\langle \emptyset, a_3 \rangle\} \tag{2.74}$$
$$\mathcal{A}^{a_3}_{in,1} = \{\langle (a_3, a_1), a_1 \rangle\}. \tag{2.75}$$

The set $\hat{\mathcal{A}}_{train}$ contains according to Eq. (2.18) only the unique articles involved in training:

$$\hat{\mathcal{A}}_{train} = \{a_1, a_2, a_3, a_4, a_5\}. \tag{2.76}$$

Computing term-weight vectors

For term selection function in Eq. (2.3), we will use a trivial term selection function σ, which selects all terms appearing in articles in $\hat{\mathcal{A}}_{train}$:

$$\sigma(\hat{\mathcal{A}}_{train}) = \{t_1, t_2, t_3, t_4, t_5, t_6\}. \tag{2.77}$$

Term t_7 was removed because it is not present in any of the articles in $\hat{\mathcal{A}}_{train}$, which contains the articles involved in training. The training setting involves slightly different set of term-weighting functions for each modality:

$$T_{out} = \langle \tau_{tf}, \tau_{idf_{all}} \rangle \tag{2.78}$$
$$T_{in} = \langle \tau_{tf}, \tau_{idf_{boa}} \rangle. \tag{2.79}$$

TF TF is an *article*-dependent weight, therefore its term-weight matrix is an $M \times N$ matrix, where $M = 5$ is the number of articles and $N = 6$ is the number of terms. The term-weight matrix for articles in $\hat{\mathcal{A}}_{train}$ and terms in $\sigma(\hat{\mathcal{A}}_{train})$ is depicted in Eq. (2.80). Comparing this matrix with Table 2.5 depicting the term weight matrix for our entire toy Wikipedia, we can observe that indeed the rows corresponding to articles not involved in training (the row for a_7) and columns corresponding to terms not returned by the term selection function (the column for t_7) are omitted.

$${}^{tf}D = \begin{bmatrix} 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 4 & 0 & 5 & 0 & 0 \\ 0 & 1 & 5 & 0 & 1 & 1 \end{bmatrix} \tag{2.80}$$

IDF-ALL The IDF-weights are *global* (refer to Subs. 2.3.4), therefore following Eq. (2.32) the term-weight matrix is an $1 \times N$ matrix, where N is the number of terms:

$$^{all}D = \left[\begin{array}{cccccc} 1.182 & 1.182 & 1.405 & 2.099 & 1.405 & 2.099 \end{array}\right]. \tag{2.81}$$

For example, the weight for term t_1 is computed according to (2.29) as follows:

$$D_{(11)} = 1 + log\left(\frac{|\mathcal{A}_{wiki}|}{|\{a \in \mathcal{A}_{wiki} : t \in a\}| + 1}\right) = 1 + log\left(\frac{6}{4}\right) = 1.182. \tag{2.82}$$

IDF-BOA The matrix for the IDF_{BOA} weight is depicted in Eq. (2.83):

$$^{boa}D = \left[\begin{array}{cccccc} 0.595 & 0.595 & 0.595 & 0.595 & 0.595 & 1 \end{array}\right]. \tag{2.83}$$

For example, the weight for term t_1 is computed according to Eq. (2.31) as follows:

$$D_{(11)} = 1 + log\left(\frac{|C|}{|\{c \in C : \exists a \in \hat{\mathcal{A}}_c : t \in a\}| + 1}\right) = 1 + log\left(\frac{2}{3}\right) = 0.595. \tag{2.84}$$

To compute the denominator, it is necessary to determine the set of target classes C for which there is at least one article a in their BOA containing term t_1. In this case, it suffices to look at the entity articles a_1 and a_3 for both target classes. Since both contain the term t_1, the sought set has the maximal possible size and is equal to the set of all target classes $C = \{c_1, c_2\}$. This leads to the maximum value of the denominator and the lowest attainable IDF.

Aggregation

The aggregation step as described in Subs. 2.2.3 requires the arrays of weights W_m, $W_{m,l}$, $W_{m,l,t}$. For this example, these weights were specified manually and are presented in Table 2.6. Sec. 2.7 describes an algorithm for learning these weights automatically from data.

As the term weight aggregation function *aggreg* we will use weighted geometric average according to Eq. (2.44). Note that the parameters in

Creating *ml*-bands According to Eq. (2.64) there are three levels for the out-link modality and according to Eq. (2.78) there are two term-weighting functions for the out-link modality. Using Eq. (2.36) we get the six corresponding term-weight vectors:

$$\beta^{TWF}_{out,0,all}(a_1) = \quad \beta^{TWF}_{out,1,all}(a_1) = \beta^{TWF}_{out,2,all}(a_1) = {}^{all}D = \left[\begin{array}{cccc} 1.182 & 1.182 & \ldots & 2.099 \end{array}\right] \tag{2.85}$$

$$\beta^{TWF}_{out,0,tf}(a_1) = {}^{tf}D_{(1*)} = \left[\begin{array}{cccccc} 1 & 2 & 0 & 0 & 0 & 0 \end{array}\right] \tag{2.86}$$

$$\beta^{TWF}_{out,1,tf}(a_1) = ({}^{out}B)^{tf}D_{(1*)} = \left[\begin{array}{cccccc} 1 & 1 & 2 & 0 & 1 & 0 \end{array}\right] \tag{2.87}$$

$$\beta^{TWF}_{out,2,tf}(a_1) = ({}^{out}B^2)^{tf}D_{(1*)} = \left[\begin{array}{cccccc} 2 & 8 & 0 & 10 & 0 & 0 \end{array}\right]. \tag{2.88}$$

Note that since IDF_{ALL} is a global term-weight (see Subs. 2.3.4) the result of Eq. (2.85) is a single term-weight vector for all levels and its computation does not take into account the incidence matrix B. On the other hand, the term-frequency, which is *article-type* term-weighting function, takes into account the incidence matrix B with the exception of the special

$W_{m,l,t}$				$W_{m,l}$			W_m	
m	l	t	value	m	l	value	m	value
out	0	TF	0.3	out	0	0.5	out	0.4
out	0	IDF_{ALL}	0.7	out	1	0.4	in	0.6
out	1	TF	0.4	in	0	0.5		
out	1	IDF_{ALL}	0.6	in	1	0.5		
out	2	TF	0.5					
out	2	IDF_{ALL}	0.5					
in	0	TF	0.6					
in	0	IDF_{BOA}	0.4					
in	1	TF	0.5					
in	1	IDF_{BOA}	0.5					

Table 2.6.: BOA Example – Training Weights

case for $l = 0$ in Eq. (2.86). In this case, the frequencies in the term-document matrix for the entity article a_1 are basically only copied. However, for $l > 0$, the $1st$ row of the corresponding power of the incidence matrix is used to denote which articles appear in the lm-band of a_1 in level l. In Eq. (2.88) and Eq. (2.87) the 1st row of the power of the incidence matrix is multiplied with the term-document matrix to get the resulting vector.

For example, the value 8 of the second component of the resulting vector in Eq. (2.88) can be interpreted as the sum of term frequencies for term t_2 in all articles in level 2. First, from Eq. (2.68) we can observe that this level contains twice article a_4. Second, the term-weight matrix in Eq. (2.80) shows that term t_2 has term frequency 4 in article a_4. Hence the value $2 \times 4 = 8$.

Applying geometric average from Eq. (2.44) as aggregator on the results of Eq. (2.85)-(2.88) we get:

$$\beta^{ML}_{out,0}(a_1) = agg(\beta^{TWF}_{out,0,all}(a_1), \beta^{TWF}_{out,0,tf}) = \left[\begin{array}{cccccc} 1.124 & 1.384 & 0 & 0 & 0 & 0 \end{array}\right] \quad (2.89)$$

$$\beta^{ML}_{out,1}(a_1) = agg(\beta^{TWF}_{out,1,all}(a_1), \beta^{TWF}_{out,1,tf}) = \left[\begin{array}{cccccc} 1.106 & 1.106 & 1.620 & 0 & 1.227 & 0 \end{array}\right] \quad (2.90)$$

$$\beta^{ML}_{out,2}(a_1) = agg(\beta^{TWF}_{out,2,all}(a_1), \beta^{TWF}_{out,2,tf}) = \left[\begin{array}{cccccc} 1.538 & 3.075 & 0 & 4.581 & 0 & 0 \end{array}\right]. \quad (2.91)$$

For example, the first element $\bar{x}_1$ of $\beta^{ML}_{out,0}$ was computed as:

$$\bar{x}_1 = 1^{0.3} \times 1.182^{0.7} = 1.124. \quad (2.92)$$

where 0.3 and 0.7 are the training weights $W_{out,0,tf}$ and $W_{out,0,all}$ from Table 2.6 and the values 1 and 1.124 correspond to the first components of vectors $\beta^{TWF}_{out,0,tf}$ from Eq. (2.86) and Eq. (2.85).

The next step is the normalization and aggregation of training instances. Since there is according to Eq. (2.59) only one training instance, Eq. (2.22) applies:

$$\beta^{INS}_{out,0}(c_1) = \frac{\beta^{ML}_{out,0}(a)}{|\beta^{ML}_{out,0}(a)|} = \frac{1}{2.509}\begin{bmatrix} 1.124 & 1.384 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.93}$$

$$= \begin{bmatrix} 0.448 & 0.551 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.94}$$

$$\beta^{INS}_{out,1}(c_1) = \begin{bmatrix} 0.219 & 0.219 & 0.320 & 0 & 0.243 & 0 \end{bmatrix} \tag{2.95}$$

$$\beta^{INS}_{out,2}(c_1) = \begin{bmatrix} 0.167 & 0.334 & 0 & 0.498 & 0 & 0 \end{bmatrix}. \tag{2.96}$$

Aggregating *ml*-bands After the normalization, the *ml*-bands within one modality are aggregated according to Eq. (2.20):

$$\beta^{MOD}_{out}(c_1) = \sum_{l=0}^{L^{out}_{max}} W_{out,l}\beta^{INS}_{out,l}(c_1) = \begin{bmatrix} 0.328 & 0.397 & 0.128 & 0.049 & 0.097 & 0 \end{bmatrix}, \tag{2.97}$$

where L^{out}_{max} is 2 for the *out*-link modality.

Aggregating modalities The training configuration for the in-link involves two term-weighting functions: TF and IDF_{BOA}. For the sake of brevity, we omit the details of computation for $\beta^{MOD}_{in}(c_1)$ and give only the final result:

$$\beta^{MOD}_{in}(c_1) = \sum_{l=0}^{L^{in}_{max}} W_{in,l}\beta^{INS}_{in,l}(c_1) = \begin{bmatrix} 0.294 & 0.492 & 0 & 0.213 & 0 & 0 \end{bmatrix}. \tag{2.98}$$

The two modalities $M = \{out, in\}$ are then aggregated:

$$\beta(c_1) = \sum_{m \in M} W_m \beta^{MOD}_m(c_1) = \begin{bmatrix} 0.308 & 0.454 & 0.051 & 0.148 & 0.039 & 0 \end{bmatrix}. \tag{2.99}$$

Since the BOA similarity function is cosine similarity, the final step is the L2 normalization according to Eq. (2.51):

$$\beta_2(c_1) = \frac{\beta(c_1)}{|\beta(c_1)|_2} = \frac{1}{0.574}\beta(c_1) = \begin{bmatrix} 0.538 & 0.794 & 0.090 & 0.259 & 0.068 & 0 \end{bmatrix}. \tag{2.100}$$

For target class c_2 we give only the final term-weight vector:

$$\beta_2(c_2) = \begin{bmatrix} 0.744 & 0.437 & 0.026 & 0.146 & 0.483 & 0.012 \end{bmatrix}. \tag{2.101}$$

2.5.3. Classification

Let us consider one noun phrase "$t_5\, t_6\, t_8$" as the only unlabeled instance x_1:

$$x_1 = \langle \text{instance } 1, \emptyset, \{t_5\, t_6\, t_8\} \rangle, \tag{2.102}$$

where $\emptyset$ refers to the empty set A_{x_1} of known Wikipedia articles.

Mapping

First, the noun phrase np ="$t_5\, t_6\, t_8$" is mapped according to Eq. (2.9) with $\rightarrow_{map}$ to a Wikipedia article.

Intuitively, given the term-weight matrix in Table 2.5, there are three possibly matching articles: a_3, a_5, a_6 all contain term t_5, term t_6 is contained only in a_5 and term t_8 was not found in any article. The article a_5 is considered the most relevant since it is the only article containing more than one term. Regarding the remaining two articles: for example a_6 could be considered as more relevant than a_3 to t_5 if the term t_5 appears in heading in a_6 and in the body of a_3. Note that this is only a very rough "demo" of the ranking function. Many factors are neglected, such as the wiki markup or the importance of the article as measured by the in-degree or page rank.

Assuming that the articles are ranked in the order of relevance a_5, a_6, a_3, the ranking function returns:

$$\rho(np) = \langle a_5, a_6, a_3 \rangle. \tag{2.103}$$

Taking the most frequent sense approach to disambiguation with Eq. (2.13) we get:

$$\delta_{mfs}(np) = a_5. \tag{2.104}$$

Since there was only one noun phrase and the set A_{x_1} is empty, the result of the mapping step is:

$$\bar{A}_{x_1} = \{a_5\}. \tag{2.105}$$

Crawling

The configuration of crawling for the classification phase is to use only[9] the *in*-link modality with $L^{in}_{max} = 2$.

Computing term-weight vectors

The classification setup features only term frequency as a term-weighting function.

Aggregation

The BOA representation for the unlabeled entity is created in the same way as outlined in the training phase, therefore we give only the resulting term-weight vector:

$$\beta_2(x_1) = \begin{bmatrix} 0.135 & 0.555 & 0.611 & 0.518 & 0.141 & 0.109 \end{bmatrix}. \tag{2.106}$$

[9] The use of just one modality in the classification phase is particularly suitable when speed is important.

Evaluation

Finally, $\beta_2(x_1)$ is compared with $\beta_2(c_1)$ and $\beta_2(c_2)$ using Eq. (2.53):

$$\begin{aligned} sim(c_1, x_1) &= \beta_2(c_1)(\beta_2(x_1))^T = \\ &= 0.538 \times 0.135 + 0.794 \times 0.555 + 0.090 \times 0.611 + 0.259 \times 0.518 + 0.068 \times 0.141 \\ &+ 0 \times 0.109 = 0.712 \end{aligned} \tag{2.107}$$

$$sim(c_2, x_1) = \beta_2(c_2)(\beta_2(x_1))^T = 0.504. \tag{2.108}$$

Note that for the second target class we again give only the final result in Eq. (2.108). The most similar target class for x_1 is c_1.

2.6. Implementation

Our experimental implementation of a BOA classifier is written in Java and relies on several Java libraries and JDK 1.6. The software has no graphical front-end, and can be executed from command line.

The software uses the following external Java libraries:

- **Lucene3** is used for access to Wikipedia index. It is an open source application released under Apache license. Obtainable from `http://lucene.apache.org/`.
- **Lucene Search Mediawiki Extension** is used as a command-line utility to generate the Lucene index from a Wikipedia dump and also for resolving noun phrases to Wikipedia articles.[10] It is an open source application released under GPL. Obtainable from `http://www.mediawiki.org/wiki/Extension:Lucene-search`.
- **Log4J** is used for logging events.[11] It is an open source application released under Apache license. Obtainable from `http://logging.apache.org/log4j/`.
- **Matrix Toolkits Java** is used to perform vector operations. It is an open source application released under GNU Lesser GPL. Obtainable from `http://code.google.com/p/matrix-toolkits-java/`.
- **JWordnetSim** is used for WordNet term weighting. It is an open source application released under GNU GPL v2.
  Obtainable from `http://nlp.shef.ac.uk/result/software.html`.
- **JWNL** (Java WordNet Library) is used as an underlying WordNet API for JWordnetSim. It is also used for lemmatization, and as a positive term list. It is an open source application released under BSD license. Obtainable from `http://sourceforge.net/projects/jwordnet/`.
- **JWSL** is used as an alternative to JWordnetSim for term weighting. The application was provided by the authors by email and the license is not specified.

[10] This software is not used as a Java library from the BOA classifier, although this is technically feasible as shown in Subs. 2.6.1.

[11] Except for `wikiindex.estimation` package, which uses java.util.logging.

Libraries Lucene3, Log4J and Matrix Toolkits Java are mandatory, JWNL and JWordnetSim are required to provide WordNet related features and JWSL is an alternative for JWordnetSim. Abstracting from JWSL, which was included only for experimental reasons, our BOA implementation relies only on free software.

2.6.1. Ranking Function ρ

As the ranking function ρ – see Eq. (2.12), the implementation implicitly uses a composite metric, which combines text-based similarity between the noun phrase and article text and article popularity as measured by the number of in-links. In fact, the ranking is a semi-black box since it is performed by the Lucene Search Mediawiki Extension.

Unlike the Mediawiki core, which is written in PHP, this extension is a Java program. Since our implementation is also written in Java, Lucene-Search can be included in `CLASSPATH` and used directly from the source code. However, the library requires extensive initialization, which even on a fast workstation requires minutes of CPU time and gigabytes of memory.

Since our BOA implementation is intended to be relatively lightweight, a tight integration would disable this positive trait. Therefore, we opted for a loose integration, where a Lucene-Search daemon runs in a separate process and communicates with the BOA classifier via the HTTP protocol.

2.6.2. Modality Membership Function μ

As modality membership function μ_m (see Eq. (2.14) and Subs. 2.3.1), there are following options – *out-links, in-links, same category*. Outlinks and in-links are retrieved directly from the Lucene index.

For *same category*, a boolean query is issued containing all the categories in the source article. The categories are added to the query with the `BooleanClause.Occur.SHOULD` modifier, which acts like a logical OR, i.e. at least one category needs to be shared. The results are sorted according to the number of categories in common.

Listing 2.1: Code fragment for category retrieval

```
TermFreqVector tfv = IndexAccessor.getMainReader()
    .getTermFreqVector(curArticleMainIndexID,"category");
BooleanQuery categoryQuery = new BooleanQuery();
for (String term : tfv.getTerms())
{
    Term t = new Term("category", term);
    TermQuery query = new TermQuery(t);
    categoryQuery.add(query, BooleanClause.Occur.SHOULD);
}
```

Limiting the number of retrieved articles Since the number of articles retrieved by some modalities may be prohibitively high, the system features two strategies to limit the number of articles processed to an externally set threshold `maxLinksToFollow`:

- with *FirstN* article selection, the system takes first `maxLinksToFollow` articles that were returned by μ_m implementation,

- with *Most similar* article selection, the system orders the matching articles by textual similarity with the *current article* using the Lucene MoreLikeThis Extension[12] and takes the first `maxLinksToFollow` articles.

Most Similar Article Selection The steps for retrieving articles related to article a in modality μ_m are as follows:

- Lucene index document identifier for article a is assigned to `mainIndexDocID`,
- list of articles related with a in modality μ_m is saved to `posList`,
- *MoreLikeThis* query is issued against `mainIndexDocID`,
- results are filtered by `posList` and only the top `maxLinksToFollow` articles are retrieved.

Listing 2.2: Retrieving similar articles

```
MoreLikeThis likeThisByText = new MoreLikeThis(
 IndexAccessor.getMainReader());
Query likeThisByTextQuery = likeThisByText.like(mainIndexDocID);
TopDocs td = IndexAccessor.getMainSearcher()
 .search(likeThisByTextQuery, posList,maxLinksToFollow);
```

Note that MoreLikeThis has a number of parameters, which are left on their default values. One of these parameters is `MinTermFreq` with its default value 2. This setting causes that MoreLikeThis query may not provide symmetric results. Consider Example 2.6.

Example 2.6 (MoreLikeThis query not producing symmetric results). Let document a_1 have content t_1 t_2 t_2 t_4 and document a_2 content t_2 t_4 t_5 t_5 t_6 t_6. A MoreLikeThis query for a_1 as a query document will search for documents containing term t_2, since it is the only term passing the `MinTermFreq=2` threshold. Document a_2 contains term t_2, therefore it will match. The opposite direction will not work. A MoreLikeThis query for a_2 as a query document will search for documents containing term t_5 and t_6 and since a_1 contain neither of these terms, a_1 will not match.

It should be also noted that the article selection strategy (FirstN or MoreLikeThis) including the `maxLinksToFollow` parameter are configured on per modality basis.

2.6.3. Term Selection

The system allows to prune terms by negative list, WordNet-based positive list combined with lemmatization, and by their BOA frequency.

Negative List – Stop words

The system involves an option to include a stop-word list. Words on the list are removed before any processing from all articles. The default list shipped with the application was compiled from multiple publicly available stop-word lists and contains 838 unique entries.

[12] `http://lucene.apache.org/java/3_0_0/api/contrib-queries/org/apache/lucene/search/similar/MoreLikeThis.html` [Retrieved on June 11, 2012]

Lemmatization

JWNL library can be used to perform lemmatization. If this is enabled, the system looks up every term encountered and creates an internal map, which replaces the term with its WordNet lemma (see Listing 2.3).

Listing 2.3: Lemmatization with JWordnetSim

```
MorphologicalProcessor morph = dic.getMorphologicalProcessor();
IndexWord w;
try
{
       w = morph.lookupBaseForm( POS.VERB, word );
       if ( w != null )
              return w.getLemma().toString ();
       w = morph.lookupBaseForm( POS.NOUN, word );
       if ( w != null )
              return w.getLemma().toString();
       w = morph.lookupBaseForm( POS.ADJECTIVE, word );
       if ( w != null )
              return w.getLemma().toString();
       w = morph.lookupBaseForm( POS.ADVERB, word );
       if ( w != null )
              return w.getLemma().toString();
}
```

Positive List – WordNet

If the $discardTermsNotInWordnet$ parameter is set to *true* terms for which no lemma is found are discarded. In this case, WordNet plays the role of the positive list. The term selection using positive list can be used only in conjunction with lemmatization. The reason for tying positive term selection to lemmatization is as follows:

- since the lemma needs to be retrieved in any case, there is minimum, if any, performance penalty for using the lemma instead of the original term,
- we expect lemmatization to have slight to negligible positive impact on the results.

Frequency-based Term pruning

Frequency-based pruning is implemented as described in Subs. 2.3.2. If the threshold is reached on a term with term frequency tf, the remaining terms with the same tf and all terms with lower tf are discarded.

2.6.4. Term Weighting Functions

In this subsection we give some remarks to the implementation of term-weighting functions discussed in Subs. 2.3.3. Most space is devoted to WordNet-based term-weighting functions, which are as far as we know unique to our work.

Our implementation does not support first sentence/paragraph boost for Term Frequency. Regarding Inverse Document Frequency, only the IDF-ALL and IDF-BOA variants are implemented.

WordNet Term Weighting

The semantic relatedness similarity measures such as Lin and Resnik that we refer to throughout this thesis as WordNet similarity measures can be adjusted to work with Wikipedia as a knowledge source, using its category hierarchy to derive the hypernym-hyponym relationship. Although using directly Wikipedia instead of WordNet as an additional resource is tempting, previous experimental results [SP06] showed that using WordNet as the knowledge source is superior to Wikipedia, we have therefore opted to use WordNet as the better performing resource.

For WordNet weight computation the measures offered by JWSL and JWordnetSim are available. For description of these two libraries please refer to Sec. 4.3. Note that these two libraries will likely produce different results with the same interest measure, which can be attributed to using different Information Content values (refer to Subs. 4.3.1 and Subs. 4.3.3).

Sense Selection The implementation does not allow to use the WordNet synset component from Eq. (2.1). The WordNet concept with which the similarity is computed is the name of the target class. In word similarity experiments (i.e. WordSim353 dataset) the concept of a target class is not present, the two words between which the similarity is computed are of equal standing. In this case, the WordNet concepts used are the names of the respective entities (entity is considered as a target entity for itself).

Aggregating WordNet measures It will be shown in Sec. 4.3 that the results of individual WordNet measures are volatile. The implementation therefore offers the possibility to use *WordNet Aggregate* measure, which allows to use multiple WordNet measures and aggregate their results using weighted arithmetic average.

We favoured arithmetic average before weighted geometric average since in our experience it often happens that one similarity measure returns zero similarity, while another measure returns non-zero similarity (also refer to Example 2.7). The use of geometric average would result in such a term obtaining zero overall similarity, which we generally consider as undesirable. It should be noted that arithmetic average is compared with the geometric average in an experiment presented in Sec. 6.2. The overall results seem to be slightly better for the geometric average.

Also, multiple WordNet measures can be used as "top-level" measures simultaneously, i.e. alongside e.g. TF and IDF. In that case, the selected aggregator (refer to Subs. 2.3.5) is applied to aggregate the WordNet term-weight vector with the remaining term-weight vectors.

Thresholds In preliminary experiments we observed that if a WordNet similarity value is in the low or high band, its precise value does not bear much significance when it comes to distinguishing the similarity of the two terms. On the other hand, the difference may be quite substantial when aggregated with values of other term-weighting functions. As a consequence, we introduced two experimental thresholds ($T_l^{low} \leq T_l^{high}$) that affect the similarity between

two terms in the following way:

$$wsim'(term_1, term_2) = \begin{cases} 0 & \text{if } wsim(term_1, term_2) < T_l^{low}, \\ 1 & \text{if } wsim(term_1, term_2) > T_l^{high}, \\ wsim(term_1, term_2) & \text{if } T_l^{low} \geq wsim(term_1, term_2) \leq T_l^{high}. \end{cases} \tag{2.109}$$

where l refers to article level and $wsim$ is a WordNet similarity measure. This threshold assumes that the maximum value of the WordNet similarity measure used is 1 and the minimum value is 0.

Example 2.7 (Aggregating WordNet measures). Consider the following configuration.

There are two target classes c_1 = basketball player and c_2 = football player. The term-weighting functions are $T_m = \langle \tau_{tf}, \tau_{JCn}, \tau_{agg} \rangle$, where τ_{tf} is term frequency, τ_{JCn} is a WordNet similarity measure Jiang and Conrath from the JWSL library and τ_{agg} is the aggregation of two implementations of WordNet similarity measure Lin (by JWordnetSim, abbreviated as JWNL, and JWSL libraries).

Assuming m refers to an abstract modality and the computation is performed on level 0, our goal is to compute the first component of $\beta_{m,0}^{ML}$, which we consider to be the weight for term t_1 ="football" for both target classes. We will denote this weight as $\bar{x}_1^{bas}$ for class c_1 basketball and $\bar{x}_1^{foo}$ for class c_2 football.

For c_1 let there be entity article a_1, and for c_2 entity article a_2. From definition, there is only one article on level 0 for entity article a_1, which is a_1.

$$\beta_{m,0}^{ML}(a_1) = agg(\beta_{m,0,tf}^{twf}(a_1), \beta_{m,0,JCn}^{twf}(a_1), \beta_{m,0,tf}^{agg}(a_1), W_{m,0,tf}, W_{m,0,JCn}, W_{m,0,agg}).$$

Let agg refer to geometric average aggregator, the first component x_1^{bas} is then computed as:

$$\bar{x}_1^{bas} = tf(t_1, a_1)^{W_{m,0,tf}} \times (jcn^{jwsl}(t_1, c_1))^{W_{m,0,JCn}} \times wagg(t_1, c_1)^{W_{m,0,agg}},$$

where the value of WordNet aggregate $wagg$ is computed by weighted arithmetic average using weights $W_{agg,lin,jwsl}$ and $W_{agg,lin,jwnl}$:

$$wagg(t_1, c_1) = W_{agg,lin,jwsl} \times lin_{jwsl}(t_1, c_1) + W_{agg,lin,jwnl} \times lin_{jwnl}(t_1, c_1).$$

Assume that the term frequency for t_1 in a_1 is 1 and 5 in a_2, $W_{m,0,tf} = 0.3, W_{m,0,JCn} = 0.2, W_{m,0,agg} = 0.5, W_{agg,lin,jwsl} = 0.6, W_{agg,lin,jwnl} = 0.4$. Note that the $W_{m,0,*}$ weights sum to 1 and also $W_{agg,*}$ weights sum to one.

Referring to Table 2.7 for values of WordNet similarity, we get:

$$\bar{x}_1^{bas} = 1^{0.3} \times 0.179^{0.2} \times (0.091 \times 0.6 + 0 \times 0.4)^{0.5} = 0.166$$

$$\bar{x}_1^{foo} = 5^{0.3} \times 0.231^{0.2} \times (0.097 \times 0.6 + 0 \times 0.4)^{0.5} = 0.291.$$

The fact that these thresholds are set separately for each level allows to apply benevolent thresholds for lower article levels and stricter threshold for higher article levels. Articles on higher levels can be expected to contain more noise as they are more distantly related to the target class. The effect of a stricter threshold is that only words closely related to the entity are included with nonzero weight.

	JWSL			JWordnetSim			all
2nd concept/similarity	Lin	JCn	*avg*	Lin	JCn	*avg*	*avg*
× basketball player	0.091	0.179	0.135	0.0	0.046	0.023	0.084
× football player	0.097	0.231	0.161	0.0	0.059	0.030	0.095

Table 2.7.: Similarity values between "football" and "basketball_player" and "football_player" for Lin and Jiang&Conrath (JCn) similarity measures as implemented in JWSL (MFS assumption, IC values computed over BNC with Resnik counting and smoothing) and JWordnetSim.

It should be emphasized that this weight adjustment is performed independently for each target class (which would be in the place of $term_2$ in Eq. (2.109)). Also note that if thresholds are used in conjunction with a WordNet aggregation measure, the values are first aggregated and then the thresholds are applied.

2.6.5. BOA Similarity Measure *sim*

The term vector computations are done with Matrix Toolkits Java library.

2.6.6. Lucene Index

A BOA classifier requires a Wikipedia index containing the following pieces of information about each article:

- term vectors with term frequencies,
- out-links, in-links, categories,
- popularity ranking (for most frequent sense relevance ranking).

Given the current size of English Wikipedia and the fact that it is constantly updated, meeting these data acquisition requirements would result in a considerable engineering effort and in fact a reimplementation of existing software as these functions are from the most part performed by the existing Lucene-Search Mediawiki Extension. The Lucene-search extension provides also the wikitext parser.

Index Description

The Lucene-based Mediawiki search engine indexes the Mediawiki article database and creates several Lucene indexes: Main index, Headlines index, Links index, Related index and Spellcheck index. The BOA classifier implementation uses the Main index containing term vectors and the Links index containing links leading out of each article.

Main Index This index is stored in the `wiki` directory and contains the following important fields: `title`, `key` with a numeric article identifier, the term vectors are saved in the `contents` field, `category` stores article's categories, `related` stores titles of articles that were determined as *related* during indexing.[13]

[13] A is said to be related to B, if A links to B, and there is some C that links to both A and B (source: Lucene-Search Extension documentation).

Links Index This index is stored in the `wiki.links` directory and and contains the following fields: `article key` containing concatenated article title, `article pageid` with a unique numeric identifier that binds the entry with the main index `key` field, `links` with a list of article titles to which the article links. The index differentiates between different types of links (article/image) using a namespace (prefix), `redirect` contains the title of the article to which the current article is redirected, `rank` reflects the number of in-linking articles.

Index Use

In the BOA classifier implementation, Lucene indexes are exploited as follows.

Term vectors Indexed Wikipedia articles are stored in the `contents` field of the Main index. Since the Lucene-Search extension does not store term vectors for this field, it was therefore necessary to modify the extension with code for storing the term vectors for the purpose of the BOA classifier.

Listing 2.4: makeDocument procedure of the `WikiIndexModifier.java` was changed; underline marks the change

```
doc.add(new Field(fields.contents(),contentAnalyzer.
 tokenStream(fields.contents(),""), Field.TermVector.YES));
```

Outlinks This information can be obtained from the `links` field of the article entry in the Links index.

Categories This information can be obtained from the `category` field of the article entry in the Links index. Again, it was necessary to modify the indexing code so that term vectors are stored.

Listing 2.5: makeDocument procedure of the `WikiIndexModifier.java` was changed; underline marks the change

```
doc.add(new Field("category",
   new CategoryAnalyzer(tokenizer.getCategories(),false)
   .tokenStream("category",""), Field.TermVector.YES));
```

Popularity ranking The Lucene-Search Extension contains a search engine, which uses sophisticated relevance ranking involving the number of in-links. The BOA implementation uses the first-ranked article as the MFS baseline.

2.7. Parameter estimation

A certain disadvantage of the BOA algorithm is the number of parameters that need to be set externally (manually). Initially, we carried out small experiments to heuristically set the values for these parameters. In this section, an algorithm for learning the parameter values is described. Since this algorithm is supervised it needs a labeled dataset to operate on.

Due to the number of parameters involved and their heterogeneity we decided to use evolutionary algorithms to perform the parameter estimation. These parameters involve particularly the weights $W_m, W_{m,l}, W_{m,l,t}$ (all float variables), but also the modality crawling depth

L^m_{max} (small integer) and maximum term-vector length (integer). For WordNet term-weighting function, there are also the T^{min}, T^{max} parameters and the weights of individual WordNet weighting functions should WordNetAggregate term-weighting function be used. There are also nominal parameters, such as the choice of article selection strategy or the infocontent file.

The overall structure of the algorithm is depicted in Alg. 7. Below we use the five-step methodology introduced in the authoritative book [Ash05] to define the individual design choices relating to the setup of an evolutionary algorithm.

2.7.1. Data Structures

Our algorithm uses the complete XML configuration file (example is in the Appendix B.4), which also includes parameters which are not subject of the optimization. This file forms the basis of what is called gene in the context of evolutionary algorithms.

The selection of parameters that will be varied is determined by another file, which we call the `GAConfig` file (from Genetic Algorithm Configuration). It lists the names of the parameters, and for each parameter its type (integer, float, gaussianInteger, enum), range and context.[14]

Enum parameters The range of an enum parameter is defined through enumeration of nominal values. Value of the parameter for a new individual is initialized by randomly selecting a value from the range. If enum feature is selected for mutation, its value is changed to a randomly selected value. Since our setup does contain only relatively short enums, the likelihood of the same value being selected during mutation is relatively high.

Integer and float The range of integer and float parameters is defined by minimum and maximum value. Value of the parameter for a new individual is initialized by randomly selecting a value from the range. If integer or float feature is selected for mutation, its value is changed by adding a gaussian.

Gaussian integer The difference between *integer* and *gaussianInteger* is in the initialization of individuals: for *integer* features the initial value is chosen with uniform probability in the range of the feature, for *gaussianInteger* it is selected with the following formula

$$newValue = min(max, min + |randomGaussian \times stddev|) \tag{2.110}$$

where *randomGaussian* is a random number with a mean of 0 and a standard deviation of 1 generated by the Java Random package `nextGaussian()` method, *stddev* is an externally set standard deviation, and *min* and *max* are lower and upper bounds given for the feature.

The motivation for Eq. (2.110) is to have the values clustered along the lower bound. The reason for this is that Gaussian initialization was designed for the *maxTermVectorLength* parameter, where a lower value leads to significantly shorter processing time.

In the preamble, the `GAConfig` also specifies the general genetic algorithm setup (maximum number of generations, population size), stopping criterion (number of generations with no improvement) and some performance parameters – type of parallel execution (multi-process,

[14] Context is a technical parameter which uses a regular expression to specify a part of configuration file, within which the feature is searched and its value changed

Feature	Parent 1	Parent 2	Child 1	Child 2
TV_LinkOut_WeightFactor_level0	0.65	0.53	0.65	0.53
TV_LinkOut_WeightFactor_level1	0.83	0.16	0.83	0.16
TV_LinkOut_maxLinksToFollow	7	10	7	10
TV_LinkOut_weightingFactor	0.07	0.49	0.49	0.07
TV_LinkOut_articleSelectionStrategy	mostsim	firstn	firstn	mostsim
TV_LinkOut_aggregationType	CustAgg2	CustAgg1	CustAgg1	CustAgg2

Table 2.8.: Crossover illustration – Crossover point is at TV_LinkOut_maxLinksToFollow

multi-threaded[15]) and the maximum concurrent processes/threads. An example `GAConfig` file is listed in the Appendix B.3.

2.7.2. Fitness Function

The fitness function measures the quality of the individual [solution] using the configuration entailed by the individual. In the BOA context, the fitness is either defined as accuracy, i.e. a ratio of correctly classified instances to all unlabeled instances, or as a value of the Spearman rank correlation. Which measure will be used depends on the experiment type, implemented experiment types are listed and described in greater detail in the Appendix B.1.

2.7.3. Variation Operators

The implementation uses both two most common evolutionary operators: crossover and mutation.

Crossover

The crossover is executed on a pair of individuals. Standard single-point crossover is implemented: a crossover point is randomly selected and then the genes are copied from parent to child so that the genes before and after the crossover point come from a different parent (refer to Table 2.8).

Mutation

After the child individuals are created with crossover, they are subject to probabilistic mutation with rate α, where α is an externally set parameter. In the selected mutation scheme, for numerical features a Gaussian mutation is performed with probability α at each position. A unit-Gaussian distributed random value ($randomGaussian$) is added to the value of selected gene (feature):

$$newValue = value + randomGaussian \times stddev, \tag{2.111}$$

where the standard deviation $stddev = (max - min)/10$. If the new value falls outside the range, it is clipped. The definition of max and min is the same as in Eq. (2.110).

[15]The multi-threaded option is experimental.

2.7.4. Selecting Parents from the Population

The way the selection is performed influences the speed of convergence. Our implementation uses single tournament selection, with tournament size 4. In this setup, the individuals are randomly divided into groups of four, and the two individuals in each group with the highest fitness are selected. The two selected individuals are subject to crossover and consequently mutation with the result of producing two offspring. The offspring then replace the two least fit members in the group of their parents.

This selection method exhibits elitism – the best individual in a population is never lost, and as a consequence the maximum fitness cannot drop as evolution proceeds.

2.7.5. Termination Condition

The algorithm stops if there is no improvement in the best fitness over a period of G generations, where the number G is an externally set parameter.

Algorithm 7 Parameter estimation with genetic algorithm

Input: $PopulationSize$ – must be divisible by 4,
$masterExpConfig = \{Param = \langle name, value \rangle\}$ – a complete set of parameters for BOA classifier,
$GAConfig = \{Param = \langle$type, range, context, Gaussian [boolean]$\rangle\}$ – a set of parameters subject to optimization, *range* is either defined for *enum* type as enum of strings, otherwise by *min* and *max* values.
$ValidationSet = \{\langle term_1, term_2, sim \rangle\}$
Output: $bestInd$ – the configuration with the highest fitness on the validation dataset

Pop:=Create an initial population of size $PopulationSize$
//initialize the population
for all Ind in Pop **do**
  Ind:=$masterExpConfig$ //create a copy of the config file
  for all $Param$ in $GAConfig$ **do**
    if $gaussian = true$ **then**
      $Ind.value$:= $min(Param.max, Param.min + |randomGaussian() \times stddev|)$
    else if $Param.type = enum$ **then**
      $Ind.value$:= random value from $Param.enum$
    else
      $Ind.value$:= uniformly distrib. rand. value between $Param.min$ and $Param.max$
    end if
  end for
end for
//start evolution
$genNumber$:=0
repeat
  for all Ind in Pop **do**
    //fitness measured by Spearman correlation coefficient or accuracy
    $Ind.fitness$:=$BOAClassifier.run(Ind.generateConfig, ValidationSet)$
  end for
  $genNumber$:= $genNumber$ +1
  $NewPop := \emptyset$
  $Groups$:= randomly partition Pop into $PopulationSize/4$ groups
  for all $Group$ in $Groups$ **do**
    $ind1, ind2$:= get two individuals with highest fitness in $Group$
    $child1, child2$:= crossover($ind1$, $ind2$)
    $child1$:= mutation($child1$)
    $child2$:= mutation($child2$)
    $NewPop$:= $NewPop \cup ind1 \cup ind2 \cup child1 \cup child2$
  end for
  Pop := $NewPop$
  $maxFitness$:= getMaxFitness(Pop)
until the $maxFitness$ is not improved for $maxGen$ generations
return individual with best fitness from Pop

2.8. Contribution and Future Work

After devising the SCM algorithm, BOA was our next step in addressing the entity classification problem. Observing the good performance of Wikipedia-based THD algorithm, an SCM component, we attempted to develop a method that will derive also the similarity from the high-dimensional textual content of Wikipedia articles rather than from the hand-coded relations in the WordNet thesaurus, which the similarity measures used in SCM rely on.

The intuition behind BOA is similar to how we imagine a modern human being would proceed if presented with the task to assess the similarity of two unknown entities. Assuming access to encyclopedia is provided, the human would first look both words up. The encyclopedic definition – the meaning behind the key concepts mentioned, rather than behind all words, as well as the overall semantic interpretation – is undoubtedly an important input for the similarity computation happening in the human brain. Our method does not attempt to perform the overall semantic interpretation, however, it does work with related "concepts", which are retrieved through the various modalities. For example, concepts mentioned in the definition are obtained with the out-link modality.

Using dictionary definitions to compare two entities is not a new idea. It was proposed already in 1986 by Lesk [Les86]. This direction was revisited in 2003 by Banerjee and Pedersen [BP03], who described the Extended Gloss Overlap measure, which takes into account also definitions of related encyclopedic entries. Lesk used Oxford Advanced Learner's Dictionary as the encyclopeadic resource, while Banerjee and Pedersen used WordNet. Using Wikipedia in a similar manner was, in fact, one of the first attempts in the area of Wikipedia-based WSC. In their 2006 paper, Strube and Ponzetto [SP06] evaluated using text overlap of Wikipedia articles on the WordSim353 dataset. The resulting correlation coefficient was only around 0.20, one of the worst results among all measures. Although the best result was achieved with Wikipedia as a knowledge source (as opposed to WordNet), it was with the Leacock and Chodorow measure, which uses only Wikipedia category hierarchy to derive the hypernym-hyponym relationship and ignores the text of Wikipedia articles. Perhaps under the influence of this paper, follow-up research largely disregarded the text of Wikipedia articles defining the entity, and focused on more elaborate uses of Wikipedia: the WLM measure uses a vector of links to represent an entity, and the ESA method uses a vector of concepts (Wikipedia articles). A more comprehensive review of related research including the description of the WLM and ESA algorithms is presented in Chapter 3.

2.8.1. Contribution in the Wikipedia-based WSC Area

With BOA algorithm, we have demonstrated that only using the text of Wikipedia articles and possibly related articles can achieve correlation with human rankings exceeding 0.7, significantly surpassing the previously achieved result of 0.2 on the same dataset and similar underlying data.

This improvement in results in our opinion stems from different similarity measures being used to compare the "bags of words" in each approach. In [SP06], the authors used an adaptation of the Lesk measure, refer to Eq. (2.112), while in BOA we use either the cosine similarity or the dot product.

$$relate(t_1, t_2) = tanh\left(\frac{overlap(t_1, t_2)}{length(t_1) + length(t_2)}\right) \tag{2.112}$$

The fundamental difference between these two text comparison methods is that the *relate* measure does not use word weights, it either counts two words or phrases as overlapping or not. In contrast, in BOA we develop an elaborate weighting scheme, where multiple weight vectors are created and subsequently combined. While the bags of articles for both compared entities need to overlap in a particular word for this word to contribute to the overall similarity, the value of the contribution of the word is derived from multiple factors, including:

- IDF_{ALL} in the entire Wikipedia,
- IDF_{BOA} computed over bag of articles of classes involved in training,
- term frequency,
- WordNet similarity with the name of the entity.

All but the last measure are well known and frequently used in information retrieval. WordNet similarity algorithms were developed and used to assess semantic relatedness of two words. As far as we know their use as a term weight is novel and we consider it to be our contribution. Also, we provide a generic framework for aggregation of multiple term-weighting functions and propose new aggregation operators. Experimental evaluation presented in Sec. 6.4 shows that the performance of the newly proposed aggregators surpasses the baseline geometric average aggregator.

BOA identifies related articles and aggregates their text into one term-weight vector. This is of course not a novel idea, definitions of related words were used in a very similar context in the Extended Gloss Overlap measure as noted above. The idea of the Bag-of-Articles does not contribute any principal change or improvement over this approach, however, to the best of our knowledge, it is substantially more elaborate than what has been proposed in the WSC area so far. Specifically, with BOA we present a mathematical framework for dealing with:

- multiple modalities (with in-link, out-link and category modality implemented),
- user-defined crawling depth in each modality,
- multiple term-weighting functions per each modality.

This framework was designed with the intent to adapt to a specific characteristic of each modality. For example, it may be the case that in-links are more relevant than out-links, however, the quality of in-links may deteriorate more rapidly with increasing article level than the quality of out-links. Similarly, the good performance of a particular term-weighting function may be tied to a specific modality or crawling depth or both. To this end, we have proposed a granular weighting scheme, allowing to set weights based on term-weighting function, modality, depth of the article (article level) and phase (training/classification). This weighting scheme is accompanied by a genetic algorithm, which learns the values of the corresponding parameters from training data. Overall, we hope our framework will contribute to the advancement of the Wikipedia-based WSC area, as we are not on one hand aware of any comparable work, and on the other hand it provides proven performance benefit. The results are close to the state-of-the-art ESA algorithm. It should be emphasized that BOA uses, in essence, only *local information* (the entity article and its neighborhood), while ESA relies on an *inverted index created from the entire Wikipedia* and is therefore more resource intensive. A more detailed information on the consumption of resources is in Subs. 7.4.

2.8.2. Future Work

During the work on the algorithm and the evaluation of related work, we encountered several possible improvements and extensions. A straightforward idea is to implement additional modalities. For example, using glosses (the first paragraph of Wikipedia pages) was found to perform slightly better than the full text of Wikipedia pages in [SP06]. What is perhaps more important is tackling the disambiguation algorithm, since it needs to be adjusted to give higher preference to the first sense. Our empirical observation on both the Czech Traveler and the WordSim353 datasets indicates that the likelihood of the first sense being correct is disproportionally higher than for any other sense.

There are two possible improvements that could be applied to the use of WordNet similarity measures. First, the WordNet term-weight vector could be made more robust by using multiple synsets for the target class. The disadvantage of this proposal is a linear, but significant rise in time complexity; the computation of the WordNet term-weight vector is already the most time-intensive operation. Second, using Wikipedia as a base for computing what is now called "WordNet term-weight vector" as suggested in [SP06] would allow to use this term-weight vector also with target classes that do not have straightforward WordNet mapping.

The best performing Wikipedia-based WSC algorithms are based on other than free text Wikipedia content. Using free text alongside the non-textual features could provide results superior to any single measure. To this end, it would be necessary to extend the mathematical framework for dealing with "multi-modal modalities" – the term-weight vector would be composed of several segments of dimensions. For example, one segment would correspond to words from free text, second to links (features used in the WLM algorithm) and the third to explicit concepts (features used in the ESA algorithm).

e	Entity, a super concept for target class and unlabeled instance
c	Target class
x	Unlabeled instance
$\mathcal{A}_{wiki}$	Set of articles in Wikipedia
$A^a_{m,l}$	Set of articles in the ml-band of the entity article a
$\mathcal{A}_{train}$	Set of all entity articles
$\hat{\mathcal{A}}_{train}$	Set of all articles involved in training
$\mathcal{A}_c$	Set of entity articles given for class c
$\bar{\mathcal{A}}_c$	Set of all entity articles for class c
$\mathcal{A}^a_{m,l}$	Set of articles related to a in modality m on level l
$\mathcal{T}$	Set of all terms
T_m	Sequence of term-weighting functions for modality m
C	Set of target classes
M	Set of modalities, M_{test} – in classification phase, M_{train} – in training phase
$\mathcal{NP}_c, \mathcal{NP}_x, \mathcal{NP}_e$	Set of noun phrases given for class c, unlabeled instance x, entity e
W	Directed walk in a graph as defined by a sequence of edges
W_m	Weight for modality m, used in function β
$W_{m,l}$	Weight for level l in modality m, used in function β^{MOD}
$W_{m,l,t}$	Weight for term-weighting function t in modality m and level l, used in β^{ML}
L^m_{max}	Maximum level in modality m
$\beta(e)$	Function associating entity e with a vector of term weights
$\beta^{MOD}_m(e)$	Function aggregating levels in modality m for entity e
$\beta^{INS}_{m,l}(e)$	Function aggregating instances in modality m and level l for entity e
$\beta^{ML}_{m,l}(a)$	Function aggregating term weights for level l, modality m and entity article a
$\beta^{TWF}_{m,l,t}(a)$	Function representing the term weight function t for ml-band and entity article a
$\sigma(\hat{\mathcal{A}}_{train})$	Function associating $\hat{\mathcal{A}}_{train}$ with a vector of terms
$\rho(np)$	Ranking function, associates the noun phrase np with a vector of its senses $\vec{s}$
$\delta(\vec{s})$	Disambiguation function, selects one sense from the vector of possible senses $\vec{s}$
$\mu_m(a, a_r)$	Modality membership function
$class(x)$	Classification function, associates the unlabeled instance x with one target class
$\rightarrow_{map}$	Operator associating a noun phrase NP with a Wikipedia article
$wsim(t_1, t_2)$	WordNet similarity between terms t_1 and t_2
IDF_{ALL}	Inverse Document Frequency over the whole Wikipedia
IDF_{TRAIN}	Inverse Document Frequency over the subset of Wikipedia involved in training
IDF_{BOA}	IDF variant, where all articles in BOA of a class are considered as one document
${}^{m}B$	Wikipedia incidence matrix for modality m
${}^{all}D$	Term weight matrix for the IDF-ALL term-weighting function
${}^{boa}D$	Term weight matrix for the IDF-BOA term-weighting function
${}^{tf}D$	Term weight matrix for the TF term-weighting function
$G = (V, E)$	Graph defined by the set of vertices V, and set of edges E

Table 2.9.: Used notation

3. Related Work

There is a large body of work related to the topic of this dissertation. The entity classification task appears in practice in the image/video retrieval, when textual annotations are used to improve the results of content-based image classifiers. Some of the work from this context is presented in Sec. 3.1. Abstracting away from the image use case scenario, the entity classification task is quite close, but not exactly fitting, any of the mainstream NLP disciplines:

Named Entity Recognition (NER) is a long established discipline which aims at classifying NEs to a predefined set of classes.[1] Large labeled corpora available for this task are exploited by supervised NER systems to learn statistical classification models. However, this approach cannot be utilized in the generic entity classification case due to the data acquisition bottleneck [CV05b]. According to survey [NS07], it is only relatively recently that semi-supervised and unsupervised approaches (also called *open domain*) to NER surfaced. Interestingly, three of five unsupervised methods mentioned in the survey are based on similar methods that we employ in SCM and THD algorithms: hypernym discovery with Hearst patterns is used in [CV05b, Eva03] and WordNet in [AM02].

Entity classification can be also cast as a **Word Sense Disambiguation** (WSD) problem if we approach target classes as word categories [FH02]. However, many WSD algorithms including [FH02] are supervised, which is not desirable in entity classification for the same reason as with NER. Further, WSD algorithms are typically constrained to finding the most fitting combination of word senses only within a local context typically limited to a window of several words before and after the entity. This is not suitable for textual annotations of objects, which are often too short to contain a usable local context.

The problem of classifying entities appearing in the text into a user-defined set of classes without the general availability of immediate left and right context for each entity can be perhaps best seen as a special case of computing semantic relatedness of individual words or noun phrases, i.e. the **Word Similarity Computation** (WSC) problem. This task is inherently unsupervised as there is no concept of a target "class", "tag" or "category" as in the NER or WSD disciplines. Both words, the similarity of which is assessed, are on equal standing and the computation result is not a crisp similar/dissimilar value but a similarity score. As a consequence, the semantic similarity computations are reliant either on semantic thesauri such as WordNet or on vast semi-structured textual resources such as Wikipedia rather than on training data.

Algorithmically, Semantic Concept Mapping (SCM), the first approach to entity classification proposed in this thesis, draws from the well-established area of WordNet-based WSC. The research in this area is closely tied with WordNet as the knowledge source,[2] therefore it seemed more logical to place the discussion into Chapter 4, which covers WordNet.

THD uses lexico-syntactic patterns to extract hypernyms from Wikipedia. Here, the research on lexico-syntactic patterns can be easily decoupled from the lexical resource (Wikipedia), because free text is typically used as input. The most influential approaches from this area

[1] Typically PERSON, LOCATION, ORGANIZATION, MISC in the CONLL task: www.cnts.ua.ac.be/conll/

[2] Most approaches to measuring semantic relatedness use WordNet as the primary knowledge source [SP06].

are recounted in Sec. 3.2. The role of Wikipedia is put to scrutiny in the dedicated Chapter 5.

The algorithms proposed in this dissertation share some of the principles with the WikiRelate!, Explicit Semantic Analysis (ESA) and the algorithm proposed by Milne and Witten that come from the niché but quickly developing area of WSC over Wikipedia. These algorithms are reviewed in Sec. 3.3.

The goal of this dissertation here is also quite close to those of Named Entity classification and Recognition, Word Sense Disambiguation. Since the work in the areas of NER and WSD is not only quite broad, but also has been previously well documented, we will by a large part skip detailed discussion of these areas and focus the discussion on a selected related algorithms in Sec. 3.4. For a more general and complete overview we refer the reader to [NS07] and [Agi07].

Sec. 3.5 shows that under certain assumptions, the problem addressed in the BOA classifier, the second approach to entity classification proposed in this thesis, can be cast as a document classification task. Sec. 3.6 discusses the influence of related work on the design of our BOA and SCM algorithms.

3.1. Image Caption Analysis

This section presents a selection of papers that report using free-text image annotations as an aid for image classification.

3.1.1. Named Entity Recognition for Visual Object Annotations

According to [DM07] the earliest system was NameIt! [SNK99], which associated names with faces in news video using the analysis of video captions and extraction of named entities from transcripts.

The system computes a composite score for each word in the transcript:

- *grammatical score*: 1.0 to proper nouns, 0.8 to common nouns, 0.0 otherwise,
- *lexical score*: 1.0 to persons, 0.8 to social groups, and 0.3 otherwise,
- *situational score*: 1.0 to speech, 0.8 to attendance at meetings, and 0.3 otherwise,
- *positional score*: 1.0 to words in the first sentence, 0.5 to words in the last sentence of a paragraph, and linearly interpolated score according to the position of the containing sentence otherwise.

The final score was computed as a product of the individual scores.

The performance of the linguistic component of their system is demonstrated on the analysis of a transcript containing 3,462 words of a 30-minute news video. The authors identified 105 name words from the transcript, the system automatically extracted 752 words as name candidates: 94 correct, 9 missed, and 658 false alarm. Due to the excessive false alarm rate the performance of named entity extraction was poor, but the overall results were promising with 33% accuracy of name-to-face retrieval.

The lexical resources used were Oxford Advanced Learner's Dictionary[3] and WordNet thesaurus. For parsing, the Link Parser [ST93] was used.

[3] `http://ota.ox.ac.uk` [Retrieved on 11 June 2012]

A more recent approach to a similar task, presented in [BBEF04], already used a NER system to improve the accuracy of extraction of named entities. Interestingly, they use the NER system contained in an early version of the GATE framework [CMBT02]. The GATE framework is used in our THD algorithm, but not for NER.

They note that "incorporating a natural language model into face clustering products produces much better results than clustering based on appearance alone". Quantitatively, they experience improvement from 67% to 77%. The contribution of NER to the classification performance is not reported.

3.1.2. Assessing if Entity is Visual

The system presented in paper [DM07] determines if entities extracted from an image annotation appear in the image. The system detects and classifies all entities (not just persons) but does not work with visual information. This research can be considered as the closest one to our SCM method (refer to Sec. 1) in that noun phrases are also extracted from text and mapped to WordNet synsets. The authors use WordNet to determine whether the entity is visual, but do not perform mapping to a custom-defined set of classes. The recognition of person names is improved through a dictionary of names extracted from Wikipedia.

As a test set, they used the Yahoo! News dataset (refer to Subs. 6.1.3). All entities appearing in 100 annotations assigned to images from Yahoo! News were classified. Using a combination of a NER system and WSD package, the authors achieved an accuracy of 75.97% when classifying entities to WordNet synsets. The erroneous entity detection accounted for 32.32% of the error, 60.56% was caused by the WSD system and 8.12% by the NER package.

3.1.3. Combining Textual and Image Features for Classification of Images

There are also multiple other approaches inspired by techniques from the area of information retrieval. For example [Wes00] uses Latent Semantic Indexing to represent information coming from both image and textual analysis into one semantic space. Image annotations are represented by full-length term vectors; no NLP is performed. The authors note that Latent Semantic Indexing as a statistical technique is less useful for named entities since these tend to occur infrequently in the corpus.

Of interest is also the work of [GAS99], which combines the textual content with image features to classify images into four categories based on the text surrounding the images on web-pages. No NLP or NER was performed, and the use of textual content resulted in marginal improvement in classification to categories for which named entities were important. This can be accounted to problems with sparsity of named entities, as also marked by [Wes00].

3.2. Hypernym Discovery

Discovering hypernyms for entities appearing in the text is an important step of the SCM algorithm presented in Chapter 1. Most research has so far focused on non-statistical approaches, often relating to the lexico-syntactic patterns introduced by Hearst in [Hea92]. Lexicosyntactic patterns for hypernym discovery are after the author of this paper often referred to as "Hearst patterns". The prototypical Hearst pattern goes along the sentence frame H0:

"An L_0 is a (kind of) L_1" (H0).

Here, L_1 is a *hypernym* of L_0, which is said to be a *hyponym*. This pattern is given along with six other patterns in [Hea92].

The Hearst pattern used for experiments was H1:

$$NP_0 \text{ such as } \{NP_1, NP_2, \ldots, (\text{and}|\text{or})\ NP_n\} \text{ (H1)},$$

where NP_0 is the hypernym and $NP_1, \ldots, NP_n$ are the hyponyms[4]. The experiments were performed on the Grolier's American Academic Encyclopedia. Hearst recalls that in the 8.6 million words 7,067 sentences were found to contain "such as" contiguously, but only 152 contained unmodified hypernym and hyponym. Out of the 156 relations 61 were confirmed (they were contained in WordNet).

The THD algorithm presented in Sec. 1.3 comes out of the research of Hearst in two ways. The hypernyms are extracted also from an encyclopedia and a hand-crafted variation of the H0 pattern is used. There are also marked differences: Wikipedia is not only significantly larger with the number of *articles* at 3.6 million (as of April 2011) at nearly half of the *words* of the Grolier's encyclopedia, but it also seems to be more suitable for extraction of H0 patterns than the Grolier's encyclopedia was for extraction of H1 pattern. We discuss the suitability of Wikipedia for hypernym discovery in greater detail in Chapter 5, some justification for this point can be found also in [LLM11].

3.2.1. Learning Lexicosyntactic Patterns

In the original paper [Hea92], Hearst does not report performing any linguistic preprocessing, which limited the possible intricacy of the proposed patterns. There was a range of follow-up research. Perhaps the most notable[5] recent work in the area of learning lexico-syntactic patterns was done by Snow et al [SJN05], who proposed an algorithm for automatically learning hypernym (is-a) relations from the text.

Algorithm

Instead of relying on hand-crafted patterns used by Hearst and many others including our THD algorithm, Snow proposed a semi-supervised algorithm for learning these patterns. The outline of this algorithm adapted from [SJN05] is given as Alg. 8. The input pairs were identified in the corpora with WordNet.

Alg. 9 illustrates how the patterns will be applied to decide if a noun pair is in a hypernymy/hyponym relationship or not.

Results

Although Snow et al. reports results in terms of F-Measure and the results of the THD algorithm is reported in terms of accuracy,[6] the best F-Measure on the hypernym task achieved

[4] Actually, a restricted version of this pattern was used that allowed only for unmodified nouns.

[5] The follow-up paper [SJN06] received the best paper award at COLING/ACL, 2006.

[6] We do not give precision, recall and F-measure for our result since it is not clear whether to count an incorrect hypernym as false positive (the system gave a wrong answer) or false negative (there was a good answer).

Algorithm 8 Learning lexico-syntactic patterns

Input: NP – list of noun hypernym-hyponym pairs $NP[i] = \langle n_1^i, n_2^i \rangle$,
Output: $classifier$ – a hypernym classifier

sentences := ∅
for all $\langle n_1^i, n_2^i \rangle$ in NP **do**
  sentences := *sentences* ∪{ sentences in which both nouns n_1^i and n_2^i occur }
end for
patterns:=parse *sentences*, and automatically extract patterns from the parse tree.
return $classifier$:= train a hypernym classifier from *patterns* as features.

Algorithm 9 Applying lexico-syntactic patterns

Input: $\langle n_1, n_2 \rangle$ pair of nouns, *testset*, $classifier$ trained classifier
Output: $result = true/false$ – noun pair is in the hypernym/hyponym relation or not

$features_{1,2}$:= extract features from n_1, n_2 in the *testset*
return $classifier$.classify($features_{1,2}$)

by Snow et al. was 0.3592 (0.8318 being the inter-annotator agreement of four annotators, with 134 marked hypernym pairs).

Discussion

Interestingly, the learning algorithm rediscovered some of the previously known patterns including H0. Snow et al use WordNet to automatically tag noun pairs with hypernym-hyponym relation (and assign the individual words the corresponding roles). The input text comes from the TIPSTER and TREC 5 newswire corpora [Har92] and later experiments were performed on Wikipedia. While this algorithm can be considered as the state-of-the-art, the approach for hypernym discovery used in this thesis takes the older hand-built route. The reason for this is that we aim to extract only the hypernym for the topic of the article, not all hypernym-hyponym pairs as in the approach of Snow et al. As will be shown in Chapter 5, the hand-crafted rules seem to work unexpectedly well on this task.

3.2.2. Hypernym Discovery using a JAPE grammar

The THD algorithm presented in Chapter 1 is closest to the research of Cimiano et al [CV05a], who use lexico-syntactic patterns also codified in a JAPE transducer grammar. JAPE (Java Annotation Pattern Engine) grammars [CMT00] were designed within the scope of the GATE Project[7] to perform finite state transduction over annotations created in a linguistic preprocessing step based on regular expressions. Though the focus of their framework Text2Onto is different as it tries to learn the whole ontology, while the work presented here tries to discover only hypernyms for the given query, it still technologically constitutes one of the most relevant instances of systems related to THD.

[7] `http://gate.ac.uk` [Retrieved on 11 June 2012]

Workflow

Linguistic preprocessing in Text2Onto is done by tokenizer, sentence splitter and POS tagger. This is the same stack of GATE NLP tools as in the THD system (see Fig. 1.2) – with the exception of noun phrase chunker, which is missing in Text2Onto but present in THD. After the linguistic preprocessing is done, the system applies JAPE patterns to extract various concepts and relations that are either to be added to the ontology or are input for another algorithms.

Extraction Grammar

Example of a JAPE grammar given as example in the Text2Onto paper [CV05a] for Hearst patterns (H1) is depicted on Alg. 10.

Algorithm 10 JAPE Grammar in Text2Onto

```
( NounPhrase1 ) : superconcept
(
{ Token.kind == punctuation }
)?
{ SpaceToken.kind == space }
{ Token.string == "such" }
{ SpaceToken.kind==space }
{ Token.string=="as" }
{ SpaceToken.kind==space }
( NounPhrasesAlternatives ) : subconcept
: hearst1
 ⟹
: hearst1.SubclassOfRelation = {rule = "Hearst1"}
: subconcept.Domain= {rule = "Hearst1" },
: superconcept.Range= {rule ="Hearst1" }
```

A detailed description of JAPE grammar syntax is available in the user manual available at `http://gate.ac.uk`.

3.2.3. Hypernym Discovery from Wikipedia

Hypernym discovery from Wikipedia is a small application area with few available papers.

Hand-built grammars

Paper [KT07] explores the use of Wikipedia as external knowledge to improve NER. Their method retrieves the corresponding Wikipedia entry for each candidate word sequence by comparing the word sequence with Wikipedia article title. The candidate word sequences are all sequences of no more than eight words that start with a word containing at least one capitalized letter.

Then they perform POS tagging and noun phrase chunking on the first sentence of the article. The last word in a noun phrase after the first “is”, “was”, “are” or “were” in the sentence was extracted. In case of the first noun phrase finishing with “one”, “kind”, “sort” or

"type" or with "name" followed by "of", they used a second noun phrase. It follows from the statistics on the occurrence of individual patterns given in the paper that the significance of skipping "kind", "sort" and "type" is negligible, with only 56 cases (altogether) out of 23,885 successful extractions.

The extracted noun can be considered as a hypernym for the candidate and is used as a feature in a Conditional-Random-Fields(CRF)-based named entity tagger.

The authors do not report on the number of correct and incorrect hypernyms and conclude that while they achieved an improvement on the NER task by using the extracted hypernyms as features of Conditional-Random-Fields (CRF) NER tagger, they believe that the hypernyms extracted from Wikipedia are too fine-grained for the classical NER task.[8] The authors conjecture that a higher impact can be achieved if the extracted hypernyms, called categories in [KT07], are used in a fine-grained NE task (such as the one proposed by [SSN02]).

Supervised learning

Paper [LLM11] presents an algorithm for extraction of hypernyms from Wikipedia with sequential supervised learning approach that combines syntactic and lexico-semantic information. As a learning data, they used 70.000 entries automatically annotated from Wikipedia. Their results confirm the very good performance of hand-built lexico-syntactic patterns with the supervised system surpassing the hand-built patterns only by a thin margin in terms of F-measure (0.851 vs 0.918).

Discussion

The THD algorithm addresses some of the issues encountered in [KT07], particularly, we used a more sophisticated extraction grammar, the search for a corresponding Wikipedia article involved also the article's popularity ranking and text, relaxed the requirement for strict match between the article title and the named entity name by utilizing string similarity functions, increased the scope of extraction from the first sentence to the lead (introductory) section of the article and as candidate word sequences we used noun phrases. Another difference is that the extracted hypernym is used in a WordNet-similarity-measure-based classifier not a CRF tagger.

3.2.4. Statistical Approaches

While purely statistical approaches such as Latent Semantic Indexing [Dee88] are prevalent in other fields of natural language processing, until recently they were only suitable for discovering symmetrical relations between words. The closest task to hypernym discovery mentioned in the seminal text book on statistical natural language processing [MS92] is unsupervised disambiguation, in which k meanings of a term are determined automatically. This approach differs in that meaning is not represented by a single word (term) but by a context. Recent research [BDMP06] introduced one of the first statistical methods for hypernym discovery. Their work utilizes Principal Component Analysis for discovering term taxonomies (hierarchies of hypernyms).

[8] I.e. classification to PERSON, LOCATION, ORGANIZATION and MISCELLANEOUS categories.

3.3. Wikipedia-based Word Similarity Computation

There is a quickly increasing body of research exploiting Wikipedia for various NLP tasks. One of the closest approaches to the BOA entity classification proposed in Chapter 5 constitute the systems WikiRelate! [SP06] and Explicit Semantic Analysis (ESA) [GM07]. The purpose of this section is not only to introduce the related approaches but also to compare them with the ones devised within this dissertation.

3.3.1. WikiRelate!

Given a pair of words w_1 and w_2, WikiRelate! searches for Wikipedia articles p_1 and p_2 that respectively contain w_1 and w_2 in their titles. Subsequently, the system extracts the categories these pages belong to and compute the relatedness based on pages extracted and the paths between the categories in the Wikipedia category taxonomy [SP06].

Semantic Relatedness

The authors of WikiRelate! implemented multiple existing semantic relatedness measures in order to see which of the measures would be best performing. All of the measures selected were traditionally used with WordNet as a knowledge source, the authors' contribution was to propose how to adapt these measures for use with Wikipedia.

- *Edge-based Measures*: Path, Leacock&Chodorow and Wu&Palmer measures (Subs. 4.2.1) computed over a taxonomy created from article categories.
- *Information Content-based Measures*: a variation of the Resnik measure (Subs. 4.2.2), which uses intrinsic information content (Subs. 4.2.4) computed from Wikipedia category hierarchy.
- *Content-based Measures*: computed over texts of p_1 and p_2 with a derivation of the Extended Lesk Overlap (Subs. 4.2.3) algorithm applied on a) first paragraphs of Wikipedia articles (*gloss*) and b) the entire text of Wikipedia articles (*text*).

Disambiguation

The system also performs disambiguation by choosing an approach which maximizes relatedness. The disambiguation is performed only if a Wikipedia search yields a disambiguation page. If a disambiguation page for w_1 is hit (lets call this disambiguation page p_1'), they first get all the hyperlinks in the page p_2'' obtained by querying for w_2 without disambiguating. The word w_2 and all the anchor text of internal links from p_2'' is purpose-tokenized and stored into lexical association list L. If a link contained in p_1' shares a token with L, it is selected and its target page becomes p_1. If there is no match, the first link in p_1' becomes p_1.

Results

Experiments were performed with all of the measures on Wikipedia and WordNet as a knowledge source. The largest dataset was the WordSim353 dataset (see Subs. 6.1.5). The performance was measured with Pearson product-moment correlation coefficient between the relatedness measure scores and the corresponding human judgments.

The best result using a single measure was obtained with Wikipedia using the Leacock&Chodorow measure (0.48). An SVM trained with the results of all measures applied both on WordNet and Wikipedia achieved an 0.59 on a test part of the WordSim353 dataset.

Discussion

The worst result achieved with the WikiRelate! method was with Extended Lesk Overlap over Wikipedia (0.19 with full-text and 0.20 with glosses). This result suggests that the hyperlink structure of Wikipedia is essential for the WSC task. On the other hand, it was shown in [MW08] that relying purely on the hyperlink structure and disregarding the category hierarchy and textual content, it is possible to achieve 0.69 correlation on WordSimilarity-353.

The primary source of information for WikiRelate! is the membership of articles into categories. This information is also exploited, albeit differently, in our BOA algorithm through the Category modality.

3.3.2. Wikipedia Link Measure

Milne and Witten [MW08] proposed an approach that uses the link structure of Wikipedia rather than its category hierarchy or textual content. The method uses anchor text to identify candidate articles for query terms. An example for two entities is given by Fig. 3.1. A public implementation of the Wikipedia Link Measure is available in the WikipediaMiner toolkit [Mil09].

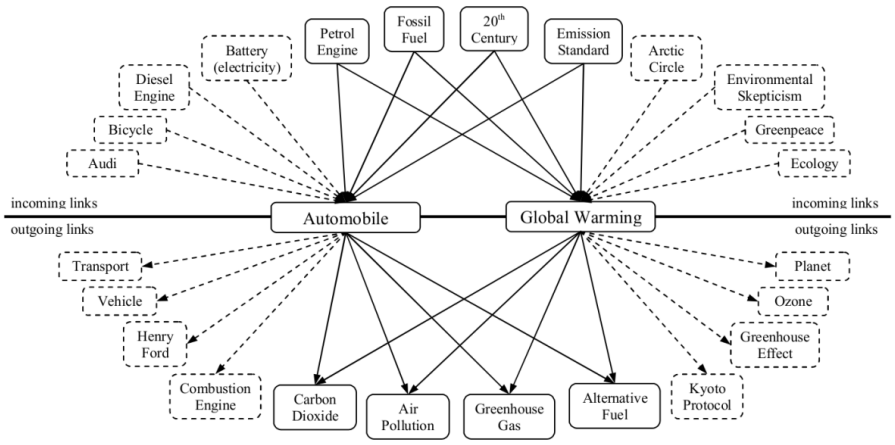

Figure 3.1.: Obtaining semantic relatedness between Automobile and Global Warming through Wikipedia Links. Source: [MW08].

Semantic Relatedness

Once the two input query terms are mapped to Wikipedia articles a and b, the relatedness is derived from the similarity of these two articles. The authors experimented with two ways of representing article content.

	Text	Corpus	Thesaurus		Wikipedia			Web
	text	LSA	WordNet	Roget	Wikirelate!	ESA	WLM	SUP
accuracy	0.19	0.56	0.35	0.55	0.48	0.75	0.69	0.78

Table 3.1.: Results for various algorithms on WordSim353 adapted from [MW08, GM07]. The best result for each algorithm. The text refers to Wikipedia article text, LSA [FGM+02], WordNet [Jar03], Roget [Jar03], WikiRelate! [SP06], ESA [GM07], WLM [MW08], SUP – supervised combination of several systems [AAH+09].

The first attempt to define Wikipedia Link Measure (WLM) draws from the TF-IDF vector space model. The key difference is that the authors measure the cosine similarity between vectors of *links* in a and b. The link counts are weighted by the probability of each link occurring. With s being the source article, t the target article, W the set of all articles in Wikipedia and T the set of all articles that link to t, then the weight of the link from s to t is expressed in the Eq. (3.1).

$$w(s \to t) = \begin{cases} log(\frac{|W|}{|T|}) & \text{if } s \in T, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

It follows from Eq. (3.1) that links are considered less significant if many other articles link to the same target. The authors give the following example [MW08]: "The fact that two articles both link to science is much less significant than if they both link to a specific topic such as atmospheric thermodynamics."

The second attempt, given in Eq. (3.2), is here based on an adaptation of the Google Distance [CV07]. While the original name of the distance measure comes from the use of the Google search engine to retrieve pages containing the terms of interest, this measure is based on Wikipedia's links rather than on Google search result. Expanding on the notation introduced earlier, A denotes the set of articles that link to a and B denotes the set of articles that link to B:

$$sr(a, b) = \frac{log(max(|A|, |B|)) - log(|A \cap B|)}{log(|W|) - log(min(|A|, |B|))}. \tag{3.2}$$

Disambiguation

Since there might be multiple matching candidate articles for given query term, the method involves a disambiguation step. The disambiguation is performed either using the most frequent sense assumption, or using one word to disambiguate the other as in the WikiRelate! method (refer to Subs. 3.3.1).

Results

This method seems to offer the best balance between the complexity of the approach, its computational demands and results on the WordSim353 collection. With the best performing setup, which is an average of the TF-IDF and the Google Distance inspired weighting scheme, the method achieves 0.69 accuracy. A comparison of the results with other approaches is given by Table 3.1.

Discussion

The fact that the method works only with the link structure, as compared to full texts of Wikipedia articles, has several positive effects. It decreases the preprocessing time and also the time required to compute the similarity at query time, since the link vector representing a particular page will be extremely sparse. The sparsity of the link vector may also be a disadvantage. Example 3.1 illustrates this on comparing links to textual content as a way to measure the similarity between articles "Burkholderia phymatum" and "Carex preussii" (both species). These two articles were randomly selected from the sample of Wikipedia articles used for a different purpose in [KCN+08b].

For the WordSim353 collection, the link sparsity issue raised in Example 3.1 is not significant: the entities contained in this dataset are mostly covered with elaborate Wikipedia articles with many links.

Example 3.1 (Link and text similarity on short Wikipedia articles).

Article 1: **Burkholderia phymatum**
Fulltext: Burkholderia phymatum is a species of proteobacteria, which is capable of symbiotic nitrogen fixation with the legume Machaerium lunatum [1].
1. Vandamme P, Goris J, Chen WM, De Vos P, Willems A (December 2002). "Burkholderia tuberum sp. Nov. And Burkholderia phymatum sp. Nov., nodulate the roots of tropical legumes". Systematic and applied microbiology 25 (4): 507–12. PMID
Links: proteobacteria, nitrogen fixation, legume, PMID (pubmed ID)

Article 2: **Carex preussii**
Fulltext: Carex preussii is a species of plant in the Cyperaceae family. It is endemic to Cameroon. Its natural habitat is subtropical or tropical dry forests. It is threatened by habitat loss. M. & Cable, S. 2000. Carex preussii. 2006 IUCN Red List of Threatened Species. Downloaded on 21 August 2007.
Links: plant, Cyperaceae, endemic, Cameroon, habitat, forests, habitat loss,

Discussion
Fulltext: words in common (excluding-stop words): species
Links: The two articles have no links in common, with the shortest path between the two articles: Burkholderia phymatum -> proteobacteria -> bacteria -> cyanobacteria -> plant.

The links between articles are also exploited in our BOA algorithm through the in-link and out-link modalities. In BOA, the link is not used directly as a component of the vector describing the entity, but it is expanded to the full text of the article it is pointing at in the case of an out-link, or emerging from in the case of an in-link. For less prolific entities, we hypothesize that the number of links can drop under a "critical number" with the link vector becoming too sparse. We hypothesize that the textual content contains more information, at least for a human, to correctly judge relatedness of two articles for a substantially higher number of articles than only the links.

3.3.3. Explicit Semantic Analysis

In the ESA method [GM07], the input text T is represented as a TF-IDF term vector. For each word w_i in the input text the method uses an inverted index to retrieve Wikipedia articles

$c_1, \ldots, c_n$ containing w_i. The semantic relatedness of the word w_i with concept c_j is computed such that the strength of association between w_i and c_j is multiplied with the TF-IDF weight of w_i in T. A schematic overview of the algorithm is given by Fig. 3.2. The relatedness score for any two documents is determined by computing the cosine similarity between the vectors of document-concept semantic relatedness.

ESA has a number of a follow-up papers describing particularly its applications in various areas of information retrieval, including cross-language information retrieval (refer to [GAS11] for an overview).

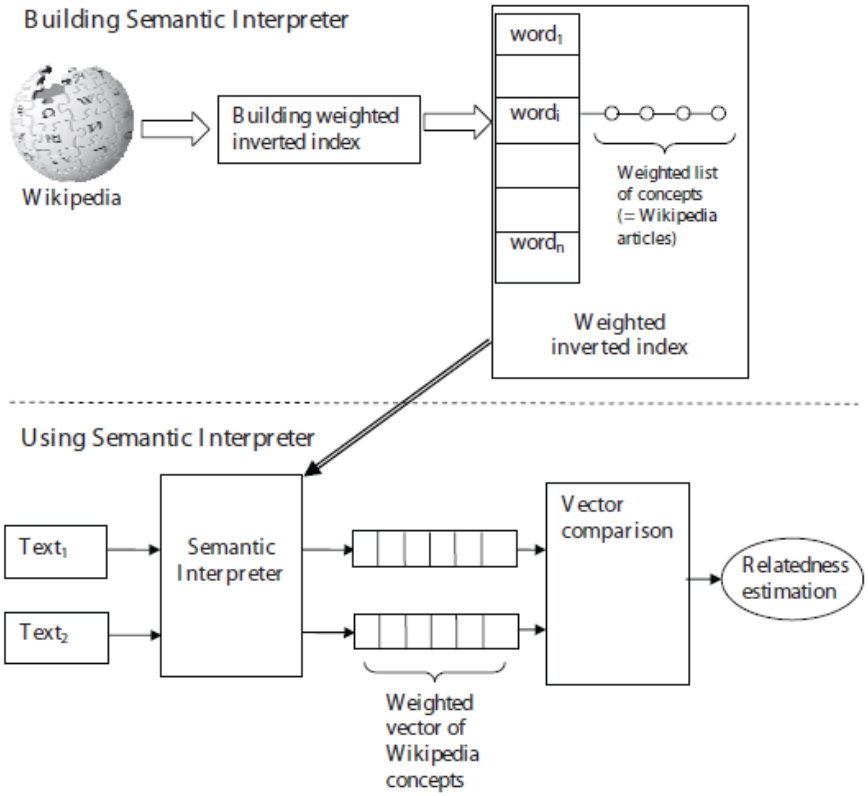

Figure 3.2.: Semantic Interpreter in ESA. Source: [GM07].

Results

The ESA method was evaluated on two tasks – individual word relatedness and document relatedness. The former task is of high relevance to the entity classification task. The result for ESA is 0.75 correlation with humans. This result, as the paper [GM07] reports, surpasses previous results including WordNet [Jar03], Latent Semantic Analysis [FGM+02] and WikiRelate! [SP06]. Better result than achieved by ESA was by [AAH+09] who achieved 0.78, but using a supervised combination of several systems. The evaluation dataset for all experiments was the WordSimilarity-353 collection [FGM+02]. For comparison with additional approaches refer to Table 3.1.

Discussion

ESA is more similar to the BOA entity classifier introduced in Chapter 2 than WikiRelate!, because it represents an entity by multiple articles from Wikipedia. However, the difference between BOA and ESA is in the way how these additional pages are identified: ESA searches the word in inverted index largely disregarding the hyperlink structure.

Another difference is in the way the semantic relatedness is assessed, while both methods use cosine similarity, BOA uses pruned[9] term-weight vectors, while ESA uses shorter document-concept semantic relatedness vectors. Shorter term vectors favour fast evaluation.

There is an interesting point related to dimensionality of the vector space used by ESA. ESA does not use the high-dimensional textual content directly, but uses it to project the article to the "high dimensional" [GM07] space of concepts – Wikipedia articles. However, this does not mean that the dimensionality of the output space is lower than for BOA or other vector-space model algorithms, for which dimensions correspond to words. For example, there are 5.5 million Wikipedia articles in the Wikipedia snapshot used in our experiments, which corresponds to the maximum number of dimensions in ESA. It is not reported in [GM07] what was the actual dimensionality of the space used in their WordSim353 experiments.

The results of ESA can be prospectively further improved, if it is combined with intensive exploitation of the hyperlink structure and categories. As was shown in Subs. 3.3.1 hyperlinks and methods using the category structure outperfom text-based methods. In this context, paper [GAS11] presents some criticism relating to the use of ESA on document collections with wide topic range such as Wikipedia. The homonyms from different topics may introduce noise and distortions, the following example is given in [GAS11]:

> "For instance, the term "capital" has several meanings (capital city, financial capital, capital letters or top part of a column). At a semantic level the meanings of this term are individually correlated with different other terms. At a syntactical level the semantic differences are lost and the correlations of all meanings with, for instance, the term "government", get blurred."

The weights in the concept vector of an entity in ESA are derived from Wikipedia articles related to words in the text of the entity article. The strength of this relationship is influenced by co-occurrences of words in the two articles. The WordNet-based term-weight vector proposed within the BOA algorithm is based on a similar idea – idea of a semantic relationship between vector components and the entity. However, the vector components are words, not articles, and the semantic relationship between the word and the entity is determined from WordNet. It could be said that ESA uses statistical techniques (specifically term correlation according to [GAS11]), while the BOA WordNet term-weight vector relies on expertly-coded relationships between words in the WordNet thesaurus. The different source of the semantic information gives an outlook for successful fusion of these two techniques in the future.

Paper [GM07] does not give enough details for verification of the results on a different Wikipedia snapshot and the authors have not released a public ESA implementation as a companion to their 2007 paper (also critically noted by [MW08]). An open C# implementation from a third party is though available [Jac07].

[9] optionally, refer to 2.3.2

3.4. Named Entity Recognition and Disambiguation

There is a large body of work on entity disambiguation, however only a limited number of algorithms exploit Wikipedia as its primary resource. A work of [BP06] is interesting in that it uses Wikipedia to generate a dictionary of named entities.

One of the most relevant of such approaches is presented in the paper entitled *System for large-scale named entity disambiguation based on Wikipedia* [Cuc07]. Below, it is described and compared to the disambiguation algorithm proposed in Sec. 2.4. Another relevant piece of work falling into this category is the system of [KT07], which was already introduced in Subs. 3.2.3.

3.4.1. Large-scale Named Entity Disambiguation Based On Wikipedia

The paper [Cuc07] uses Wikipedia as the main resource to perform name entity classification. The system tries to identify entities using four sources of information: titles of entity pages, titles of redirecting pages, disambiguation pages and references to entity pages in other Wikipedia articles. The mentions of entities in the text are called *surface-forms*, while we call them *noun phrases* in this dissertation. The system then tries to map each surface form to the corresponding *entity article*, which is a Wikipedia article focused on a single entity.

Candidate Selection

Wikipedia entity articles that have a surface form matching the disambiguated surface form are considered as candidates. Entity surface forms are created from the titles of entity pages, titles of redirecting pages and the references to entity pages in other Wikipedia articles.

Candidate Representation

For each candidate, an *entity vector* of *contexts* and *categories* is created. The *context* component contains articles to which the candidate entity article links from the first paragraph and articles for which the corresponding pages refer back to the candidate entity. The *categories* component contains category tags that are either directly assigned to the article or extracted from Wikipedia list pages it appears on.

The entity vector for a candidate entity is a 0-1 vector of length M+N, where M is the number of known contexts and N the number of known category tags. The role of the entity vector is similar to that of the BOA vector. However, the elements of the BOA vector are term weights, while the elements of the entity vector are binary values: 1 if category/context does appear for the candidate and 0 otherwise.

Discussion

In our approach, we match candidate articles by full-text search in Wikipedia, which uses a number of in-links as one of the parameters. The advantage of this approach is that it ranks first the (likely) most frequent of the candidate entities.

The way articles are selected for the context component reflects the experiments of the authors of [Cuc07] with broader inclusion strategy (e.g. all linked articles), which yielded worse result. In contrast, the recursive definition of BOA uses the broader inclusion strategy

and in addition to it, articles that are not directly linked with the candidate entity can be involved.

The argument is that an additional article might contribute a valuable information to the bag in terms of words contextually related with the entity provided it is possible to remove sufficient amount of the noise constituted by the unrelated words. The BOA approach aims to tackle the noise problem particularly through the use of WordNet similarity-based filtering and/or by using only the most frequently used words as described in Subs. 2.6.3.

The dominating effect of frequently occurring words shared by many articles, which survived through term selection, can be mitigated by the use of IDF in the term-weighting function.

The disambiguation in [Cuc07] is performed in the following way. A disambiguated document is represented with a *document vector*, which aggregates all possible entity disambiguations (their contexts and categories) of each surface form appearing in the document. This document vector is subsequently compared with entity vector of each possible entity disambiguation and the assignment of entities to surface forms that maximizes the similarity of the vectors is selected.

In Chapter 2 we suggest to use a more subtle representation against which candidate entities are compared to than the aggregate of all possible senses in the document vector is. In our approach, the candidate entity is compared with several clusters that are intended to group entities of like meaning. The number of necessary comparisons is alleviated by the initialization of sense assignment to the most frequent sense and by attempting to change the sense only for entities that do not fit well any cluster.

Unfortunately, it is not possible to compare the result of our algorithm and that proposed by [Cuc07] because [Cuc07] does not give results on a standard dataset nor is the implementation publicly available.

3.5. Document Classification

The de-facto standard for document representation in the document classification task is the word-based vector (Bag of Words, or BOW), where each dimension is associated with a term of the dictionary containing all the words that appear in the corpus. The feature weights are typically term frequency values in a given document.

The BOA approach represents both the entity and the candidate class entities as the centroid vector of the Wikipedia articles using BOW with the difference that the individual word weights are a weighted average of weights in multiple documents included in the bag. The BOA task can thus be viewed as a document classification problem if we abstract away from the problem of selecting correct articles to the bag and the process of aggregating the BOW of individual documents.

Book [CST00] defines the BOW approach as follows:

$$\phi : d \rightarrow \phi(d) = (tf(t_1, d), tf(t_2, d), \ldots, tf(t_D, d)) \in R^D, \tag{3.3}$$

where $tf(t_i, d)$ denotes the term frequency of the term t_i in document d, and D is the number of terms in the Dictionary.

There is a large body of work on document classification and it would be a futile work to even try to summarize it. After introducing the Rocchio classifier, the basic generic document classification technique, this section will focus on selected Wikipedia-based approaches to document classification. One of the most commonly criticized aspects of the BOW representation

is its inability to capture word relatedness. The Wikipedia based approaches use the hyperlink structure between Wikipedia documents to extract additional information about relatedness of concepts described by the article. The hyperlink structure was unaccounted for in older document categorization/classification research.

3.5.1. Rocchio Classifier

The Rocchio classifier [MRS08] is one of the simplest classifiers working on top of the BOW representation. This algorithm was originally proposed by Rocchio in 1971 [Roc71] for relevance feedback, but was later adapted for text categorization.

Rocchio classifier assigns documents (unlabeled instances) to one of the target classes – document categories. It is a supervised classifier. Input for the training phase is a set of documents for each target class C. The output of the training phase is a centroid for each category (refer to Alg. 11).

The test phase comprises simply of assigning the test document to the closest centroid (refer to Alg. 12). The documents $\vec{d}$ in both training and test phase are vectors of word weights.

Algorithm 11 Rocchio classifier – TRAIN

Input: C – set of J target classes, each class assigned a set of training documents
Output: $classifier$ – a set of centroids
  for all $c_j \in C$ **do**
    $D_j := \{\vec{d}:$ documents in class $c_j\}$
    $\vec{y}_j := \frac{1}{|D_j|} \sum_{\vec{d} \in D_j} \vec{d}$
  end for
  return $classifier := \vec{y}_1, \ldots, \vec{y}_J$.

Algorithm 12 Rocchio classifier – TEST

Input: $classifier = \vec{y}_1, \ldots, \vec{y}_j, \ldots, \vec{y}_J$ – learned classifier, $\vec{d}$ – test document
Output: $class$ – a class label for $\vec{d}$
  return $\arg\min_j |\vec{y}_j - \vec{d}|$

Some sources note (e.g. [Joa97]) that there are three design choices when implementing this algorithm:

- word weighting method,
- document length normalization,
- similarity measure.

Note that the Alg. 11 and Alg. 12 were adapted from the authoritative textbook [MRS08], which already made specific choices as concerns document length normalization and the similarity measure (Euclidean).

According to [Joa97] the most straightforward adaptation of the Rocchio algorithm to text categorization for domains with more than two categories is a configuration with TF-IDF for word weighting method and cosine similarity. Additionally, this adaptation supports negative instances. The prototype vector is then computed as:

$$\vec{y}_j = \alpha \frac{1}{|D_j|} \sum_{d \in D_j} \frac{\vec{d}}{|\vec{d}|} - \beta \frac{1}{D - D_j} \sum_{\vec{d} \in D - D_j} \frac{\vec{d}}{|\vec{d}|}, \tag{3.4}$$

where α and β are parameters that express the weight of positive and negative training instances, D is a set of all documents involved in training, D_j the set of documents assigned to class j and $|\vec{d}|$ expresses Euclidean length of vector $\vec{d}$.

3.5.2. BOW Enrichment

Paper [WHZ+07] shows a system for extending the BOW representation with background knowledge extracted from Wikipedia.

The basis of their approach is a thesaurus covering the hyponymy, hypernymy, associative and synonymy relations. Once the thesaurus is available, the terms in the document are mapped to concepts in the thesaurus and the neighborhood of the concept is added to the document's BOW representation.

Building the Thesaurus

The titles of (non-redirect) Wikipedia articles constitute the descriptor terms in the thesaurus, while the non-descriptor terms – the synonyms – are retrieved from redirect pages. This covers also abbreviations and common misspellings. The hyponyms are the names of subcategories of the categories the article belongs to. Disambiguation pages are used to identify polysemous concepts. The hypernyms are the names of the categories the article belongs to. Both hyponyms and hypernyms can be retrieved to multiple levels. The associative relations are selected from the articles to which hyperlinks on the page point at based on the combination of its relevance to the original article, which is computed as a combination of the following three measures: *content*, *out-link category* and *path distance*.

Assuming A is the original article and $B^A = \{B_1, \ldots, B_i, \ldots, B_n\}$ be the set of Wikipedia articles that are targets of hyperlinks from A, these measures are defined as follows:

- The *content measure* $S_{BOW}(A, B_i)$ uses cosine similarity to compute the relatedness between TF-IDF BOW representations of the articles A and B_i.
- The *out-link category measure* $S_{OLC}(A, B_i)$ uses cosine similarity to compute the relatedness between A and B_i, both represented with a vector of the categories the respective article belongs to.
- The *path distance* $D_{cat}(A, B_i)$ between A and B_i is measured as the shortest path connecting the categories the respective article belongs to normalized by the depth D of the taxonomy:

$$D_{cat}(c_1, c_2) = \frac{length(c_1, c_2)}{D}. \tag{3.5}$$

The final similarity between A and B_i is computed using the following formula:

$$S_{overall}(A, B_i) = \delta_1 S_{BOW}(A, B_i) + \delta_2 S_{OLC}(A, B_i) + (1 - \delta_1 - \delta_2)(1 - D_{cat}(A, B_i)) \tag{3.6}$$

Here, $\delta_1, \delta_2 \in (0, 1)$ are weighting constants. The paper suggests a procedure for experimentally setting the values of the first three parameters based on training data. The concepts

represented by articles in B^A that have the associated $S_{overall}(A, B_i)$ score above a predefined threshold are added into the associative relation with A into the thesaurus.

Extending the BOW

For the feature enrichment process, the original document is processed. Term sequences are extracted from the document, and then matched with thesaurus concepts (Wikipedia article titles). The words in the concept name need not follow in the document in the immediate sequence, they however need to appear within a certain window. The window size decreases with the length of the concept name in terms of the number of tokens. The candidate concepts subsumed by another candidate concepts are removed.

The (disambiguated) concept appearing in the document is called a *candidate concept*. The candidate concept and its *related concepts* (synonyms, hyponyms and associated concepts) are added to the document. The resulting BOW representation of the document is

$$\phi(d) = (\langle\text{terms}\rangle, \langle\text{candidate concepts}\rangle, \langle\text{related concepts}\rangle). \tag{3.7}$$

Disambiguation

If the concept is polysemous, disambiguation is applied. There are two disambiguation techniques: *disambiguation with text similarity* and *disambiguation with context*.

In the disambiguation with *text*, the article describing each of the applicable concepts is described using TF-IDF BOW and compared with the TF-IDF BOW representation of the document. The concept with the highest similarity is selected.

The disambiguation with *context* is based on paper [ARG95] and adapted to Wikipedia. The disambiguation works on a sentence scope. Within the sentence, the non-polysemous concepts are used to resolve the polysemous concepts.

For a polysemous concept c_i, its distance to each of the non-polysemous concepts in the sentence c_j is computed using the following formula:

$$D_{conc}(c_i, c_j) = \begin{cases} 1 & \text{if } c_i \text{ is a synonym of } c_j, \\ 1 & \text{if } c_i \text{ is an associated concept of } c_j, \\ D_{cat}(c_i, c_j) & \text{otherwise,} \end{cases} \tag{3.8}$$

where category distance is defined by Eq. (3.5). The disambiguation algorithm (Alg. 13) computes the average distance $dist_{average}$ of each sense of the polysemous concept and all non-polysemous senses within the sentence. The sense with the smallest distance is returned. If there are no non-polysemous concepts, the text similarity is used instead.

Discussion

The disambiguation algorithm used in our BOA classifier can be perceived as a mixture of the *text*- and *context*- based classifiers of [WHZ+07]. Since our basic measure of similarity is based on text, not on the category information as in the *context*- based classifier, the same approach can be applied to all entities.

The context disambiguation algorithm of [WHZ+07] reacts to context drift by limiting the scope of disambiguation to a sentence level assuming that entities within one sentence will be

Algorithm 13 Context Disambiguation algorithm

Input: m disambiguations of a polysemous candidate concept c^p: $\langle c_1^p, \ldots, c_m^p \rangle$,
n non-polysemous candidate concepts $c = \langle c_1, \ldots, c_n \rangle$
Output: disambiguation c_j, where $1 \leq j \leq m$

```
if n = 0 then
  return ∅
else
  for j=1 to m do
    dist_average[j] = (Σ_{i=1}^{n} D_conc(c_j^p, c_i)) / n
  end for
  return arg min_j dist_average[j]
end if
```

contextually related. In contrast, the BOA disambiguation algorithm performs disambiguation on a global context of all entities. The fact that there will be multiple contexts present in the input text fragment is taken into account through entity clustering.

This approach has, at least conceptually, two advantages. First, the disambiguation starts at the most frequent sense baseline, which was not until recently never surpassed using unsupervised algorithms [PDKM09]. Second, disambiguation can be performed even without any non-polysemous concepts [WHZ+07].

3.5.3. Semantic Kernels

Using semantic kernels for text classification using Wikipedia [WD08] is an extension of the approach introduced in paper [WHZ+07], which was discussed in Subs. 3.5.2.

Document Representation

Paper [WD08] takes up the BOW representation from Eq. (3.7). It contains the two subvectors with candidate concepts and related concepts in addition to terms. In this later work, the authors suggest minor changes to the way $\phi(d)$ is created. Perhaps the most marked ones are:

- terms that were mapped to candidate concepts are excluded from the term subvector,
- word sense disambiguation is computed using text similarity,
- candidate concepts are searched using exact matching strategy.

Thesaurus

This method relies on a thesaurus built-from Wikipedia in a very similar manner to the one introduced in Subs. 3.5.2. The main change is perhaps that the $S_{overall}$ in Eq. (3.6) is used not to prune the associative concepts, but in a later stage to fill the proximity matrix.

Similarity Computation

The core of the difference between the work on semantic kernels [WD08] and their earlier work [WHZ+07] is the transformation of the BOW document vector $\phi(d)$ to $\tilde{\phi}(d) = \phi(d)S$, where

S is a semantic matrix. The corresponding vector space kernel takes the form:

$$\tilde{k}(d_1, d_2) = \phi(d_1)SS^T\phi(d_2)^T = \tilde{\phi}(d_1)\tilde{\phi}(d_2)^T \tag{3.9}$$

The semantic matrix S is defined as $S = RP$, where R is a diagonal matrix containing term weights and P is a proximity matrix. The authors suggest to use Inverse Document Frequency (IDF) for term weights in R. As a consequence, matrix R is already embedded in the weighting matrix $\phi(d)$.

The proximity matrix P consists of four submatrices. The *terms* $\times$ *terms* matrix, the *concepts* $\times$ *concepts* matrix, *concepts* $\times$ *terms* matrix and *terms* $\times$ *concepts* matrix.

The crucial submatrix is the *concepts* $\times$ *concepts* matrix, the values of entries in this submatrix are defined as follows:

$$P_{ij} = \begin{cases} 1 & \text{if } c_i \text{ and } c_j \text{ are synonyms,} \\ \mu^{-depth} & \text{if } c_i \text{ and } c_j \text{ are hyponyms,} \\ S_{overall} & \text{if } c_i \text{ and } c_j \text{ are in an associative relations,} \\ 0 & \text{otherwise,} \end{cases} \tag{3.10}$$

where $S_{overall}$ is computed using Eq. (3.6). The constant μ was experimentally set to $\mu = 2$, and *depth* reflects the number of "hops" between category of page of concept i and page of concept j in Wikipedia. The minimum value is 1 (direct hyponym).

The *concepts* $\times$ *terms* and *terms* $\times$ *concepts* matrix are zero matrices (all elements zero). The *terms* $\times$ *terms* is a diagonal matrix with ones on the main diagonal.

The benefit of the semantic kernel approach is that the transformed document representation $\tilde{\phi}(d) = \phi(d)P$ is less sparse than the original $\phi(d)$, since $\tilde{\phi}(d)$ has nonzero entries for all concepts that are semantically related to d.

3.6. Impact of Related Work

This section summarizes the influence of related work on our contribution. Sec. 3.1 described the use of entity classification in image caption analysis. From this area came the motivation and subsequently inspiration for this dissertation. A vital part of our first attempt at the entity classification problem was the THD algorithm. This algorithm is an application of hypernym discovery algorithms, these are reviewed in Sec. 3.2. Sec. 3.3 reviews Wikipedia-based WSC algorithms; these are closely related to our second attempt, the BOA algorithm.

The WSC research presented in Sec. 3.3 evolved almost in parallel with this dissertation. The first approach to compute semantic relatedness measures using Wikipedia was according to [MW08] the Wikirelate! method, which appeared in the 2006 paper [SP06]. For comparison, the work on this dissertation started in 2007 with the first paper appearing in 2008 [KCN+08a]. We became aware of this line of research only in the writing up stage. It had therefore no influence on the design of neither the BOA nor the SCM algorithms. However, the standard dataset in the WSC area, the WordSim353 dataset, was used for the evaluation of our algorithms.

There is hopefully also some positive impact of our ignorance of the Wikipedia-based WSC algorithms. Our results are somewhat orthogonal to what was proposed in this area so far, which gives opportunities for cross-fertilization in the future. The design of our algorithms

was also in no way influenced by the rather specific composition of the WordSim353 test set. One could suspect this might have happened for some other algorithms, where the motivation to improve results over previously proposed algorithms on this dataset might have influenced the design. The WordSim353 dataset contains almost exclusively prolific words and no named entities. In contrast, our aim is the general entity classification task, where the entities might be single nouns, noun phrases, or named entities.

Sec. 3.4 gave a brief overview of NER algorithms. We studied some representative approaches from this area before our algorithms were designed, and we concluded that the general outline of a NER algorithm is not in principle suitable for the entity classification problem as we posed it. These algorithms require extensive number of training examples, which is not in line with our "no training set from the user" design requirement. It should be noted that this in fact follows from two other requirements: "follow the zeitgeist" and "accept user-defined set of target classes or no target classes at all" as stated in the introduction. Sec. 3.5 discussed techniques used for document classification. Rocchio classifier is perhaps the single most important piece of related work that influenced the design of our BOA classifier.

4. WordNet as a Knowledge Source

WordNet is used in both SCM and BOA algorithms. It is central to SCM as WordNet similarity measures are used to perform the entity classification. In BOA its use is optional, but more varied. It can be used as term-weighting function, as a positive term list and even as a lemmatizer.

WordNet is a large English thesaurus that was created at the Princeton University [Fel98]. It covers nouns, verbs, adjectives and adverbs. Synonyms are grouped together into *synsets*. There are 155,287 words (117,798 nouns, 11,529 verbs, 21,479 adjectives, 4,481 adverbs) grouped into 117,659 sets of synonyms (synsets).[1] For each synset there is a short dictionary explanation available called a *gloss*. There are several types of relations captured between synsets dependent on the type of the synset. These types are discussed in Subs. 4.1. The main relationship used to impose a hierarchical structure over the thesaurus is the *hypernym-hyponym* relationship. WordNet is an acclaimed lexical resource that is widely used in the literature for word similarity and word disambiguation computations. The description of the commonly used WordNet similarity measures is in Sec. 4.2; their implementations are described in Sec. 4.3. An assessment of the use of WordNet in the SCM and BOA algorithm is presented in Sec. 4.4.

4.1. Types of Relationships between WordNet Synsets

The types of relationships captured between nouns [Fel98]:[2]

- antonym: opposite of a word,
- hypernym: Y is a hypernym of X if every X is a kind of Y,
- hyponym: Y is a hyponym of X if every Y is a kind of X,
- holonym: Y is a holonym of X if X is a part of Y,
- meronym: Y is a meronym of X if Y is a part of X.

The types of relationships captured between verbs:

- antonym
- hypernym: the verb Y is a hypernym of the verb X if the activity X is a (kind of) Y,
- entailment: the verb Y is entailed by X if by doing X you must be doing Y.

The types of relationships captured for adjectives include:

[1] WordNet 3.0. source: `http://wordnet.princeton.edu/wordnet/man/wnstats.7WN.html` [Retrieved on 11 June 2012]. The statistics for previous WordNet version do not differ substantially.

[2] Or refer to the online documentation `http://wordnet.princeton.edu/man/wninput.5WN.html`

- antonym opposite of a verb,
- pertainym (pertains to noun),
- similar to,
- participle of verb.

The types of relationship between adverbs:

- antonym,
- derived from adjective.

4.2. Similarity and Relatedness Computation

Various similarity measures using WordNet have been proposed. A solid overview of the foundations of word and concept relatedness measures is given in [BH06]. The paper gives a detailed overview of methods that are based on different kinds of lexical resources (dictionary-based approaches, Roget thesauri-based, WordNet-based) and then gives an in-detail comparison of selected approaches, most of which are either directly developed for WordNet or at least use WordNet. The explanation for the apparent preference for WordNet among word-similarity researchers is that the noun network of WordNet was the first to be richly developed [BH06]. The free availability of WordNet might have also played a role.

Paper [BH06] also clarifies the terms *semantic relatedness*, *similarity* and *distance* which are often used interchangeably.

- *semantic similarity* uses the hypernym-hyponym relationship[3] and the synonymy relationship,
- *semantic relatedness* is a more general concept than semantic similarity as it involves also the antonymy, meronymy or any kind of functional relationship or frequent association,
- *semantic distance* is an inverse of semantic similarity.

The difference between semantic *similarity* and *relatedness* can be illustrated on the following example given by Resnik [Res95] and repeated by [BH06]: "Cars and gasoline would seem to be more closely related than, say, cars and bicycles, but the latter pair are certainly more similar." Technically, in the similarity computation we are primarily interested in the is-a relationship [RB89].

Abstracting away from semantic *distance* as an inverse of semantic similarity, both the semantic similarity and relatedness tasks are relevant to the methods proposed in this paper. In the SCM method presented in Chapter 1.1 the classification task can also be cast as a (noun phrase) *similarity* computation, since WordNet similarity measures are used. On the other hand, while WordNet similarity computation is also used in BOA, it is used only for term weighting; the BOA algorithm works on a statistical basis which provides too gross means to differentiate between the different kinds of relationships between words. As a consequence, the output of the BOA algorithm is somewhere between the similarity and relatedness tasks.

[3] Although this is not directly stated in [BH06], the hypernym-hyponym follows from the examples given there: bank-trust company.

4.2.1. Edge-Based Models

Edge-based algorithms use the edges linking the synsets. There is a range of ways to extend this basic principle, [JC97] gives the following list: depth of a node in the hierarchy, type of link and strength of an edge of the link.

Path Measure

The baseline distance between A and B is provided by Rada [RB89], who proposes to use the number of nodes between A and B on the shortest path connecting the two concepts c_1 and c_2 [RB89] (cited according to [SP06]):

$$rada(c_1, c_2) = length(c_1, c_2). \tag{4.1}$$

If used with WordNet, only hypernym-hyponym relationships tend to be considered [OSdCI11]. Although this is an old (published in 1989) and simple measure, experimental results indicate it is superior to other measures in some tasks. For example, in the sentence similarity task in [OSdCI11], the Path measure exceeds all other measures including Lesk, Lin, Jiang and Conrath and Resnik, all described later in this section.

Wu & Palmer

The Path measure does not take into account neither the absolute position of the lowest common subsumer (lso) of the concepts in the taxonomy, nor the relative distance of the two concepts from the lowest common subsumer. In the Wu & Palmer proposal [WP94], the concepts with a more specific lowest common subsumer are considered as more similar, and the shorter the distance between the lso and either of the concepts, the higher the similarity. The function $depth(x, y)$ denotes the number of nodes on the path between node x and node y, $root$ refers to the root node. The measure is defined as:

$$wup = \frac{2\, depth(lso(c_1, c_2), root)}{depth(c_1, lso) + depth(c_2, lso) + depth(lso(c_1, c_2), root)}. \tag{4.2}$$

Wu & Palmer measure achieves its maximum similarity value 1, if c_1 and c_2 are equal.

Leacock & Chodorow

The Path measure does not take into account the depth of the taxonomy. This is corrected by the measure proposed by Leacock & Chodorow in 1998 [LC98]:

$$lch(c_1, c_2) = -log\frac{length(c_1, c_2)}{2D}, \tag{4.3}$$

where D is the maximum depth of the taxonomy. If we assume the existence of a root node connecting all WordNet hierarchies[4] then Leacock & Chodorow measure gives the same (relative) results as the simpler and computationally more efficient path measure, since taxonomy depth D acts only as a scaling constant.

[4] Although not a part of WordNet, it is a common practice to insert a root node, for example Ted Pedersen's WordnetSimilarity online demo at `http://marimba.d.umn.edu/cgi-bin/similarity/similarity.cgi` inserts root node by default.

4.2.2. Information Content-based Approach

This class of measures is distinguished by associating an additional "significance" value with each concept in the taxonomy.

Resnik measure

This approach was suggested by Resnik [Res95] and can be expressed using the following formula:

$$sim_{res}(c_1, c_2) = IC(lso(c_1, c_2)) = -log\, p(lso(c_1, c_2)), \tag{4.4}$$

where function lso returns the lowest common subsumer from the hierarchy and the value

$$IC(c) = -log(p(c)) \tag{4.5}$$

is called Information Content (IC).

The value $p(c)$ denotes the probability of encountering an instance of concept c, which is estimated from frequencies from a large corpus:

$$p(c) = \frac{\sum_{w \in W(c)} count(w)}{N}. \tag{4.6}$$

The interpretation of this measure is following: the higher the lowest common subsumer of the two terms in the hierarchy the lower their similarity. If it is the top node then their similarity is 0. The values $p(c)$ in the Resnik's experiments was computed from the Brown corpus, which is a 1,000,000 word collection of texts across multiple genres [FK83] (cited according to [Res95]). As an occurrence of a concept (noun) c Resnik counted not only the occurrences of the word representing the concept but also of terms that were direct or indirect hyponyms of concept in WordNet. This is expressed by function $count(w)$. The set of all these nouns is denoted as $W(c)$, N denotes the total number of noun tokens in the corpus that are also present in WordNet.

Other approaches to computing information content were also suggested. For example, Pirro and Seco suggested *Intrinsic Information Content*, which is detailed in Subs. 4.2.4.

The Resnik similarity measure was criticized [BH06] for neglecting disambiguation, since towards $p(c)$ also terms unrelated to c can be counted, and for using the edges between words only to identify direct and indirect hypernyms, disregarding other information such as their distance counted by the number of edges. Another problem with this measure is that the similarity of two identical concepts is not 1, but the information content of their lowest common subsumer.

4.2.3. Gloss-Based models

WordNet synsets are described by free-text glosses. Intuitively, the semantic similarity of synsets will be correlated with the textual similarity of glosses.

Lesk Algorithm

The Lesk algorithm [Les86] was proposed to address the Word Sense Disambiguation problem by trying to select the correct combination of word senses through counting overlaps between

dictionary definitions of their various senses. Selected is that sense of the target word, the gloss of which has the most words in common with the glosses of the neighbouring words.

The original algorithm was tested on three machine readable dictionaries: Webster's 7th Collegiate, the Collins English Dictionary and the Oxford Advanced Learner's Dictionary of Current English; all three achieving comparable results. This notion was taken up by various WordNet adaptations of the Lesk algorithm.

Extended Gloss Overlaps

The shortness of the dictionary definitions is the obvious shortcoming of the Lesk algorithm. This can be addressed in WordNet by involving glosses of related synsets. This is the principle of the *Extended Gloss Overlaps* measure proposed by Banerjee and Pedersen [BP03]. This idea was taken one step further by [SP06], who adapted the Lesk measure for use with Wikipedia: text overlap is computed from the first paragraph of the text of the pages (called gloss) and full page texts. The approach of [SP06] is in greater detail discussed in Subs. 3.3.1.

Although the gloss-based measures are best-performing in some tasks (e.g. Word Sense Disambiguation in [PBP05]), they are inferior to simpler measures in other tasks [OSdCI11]. The reasons given by [OSdCI11] are:

- Similarity of two identical words depends on the word, because all the senses associated to each word are given. The authors give an example for the extended gloss overlap measure, which gives a similarity of 703 to a pair of words "chicken" and "chicken", but similarity 2529 to the pair of words "dog" and "dog".
- Gloss-based measures give different similarity for synonyms, because words within the same synset have different associated glosses.
- Gloss-based measures allow for comparison of words that play role of different parts of speech. In the authors' experience, it is better to avoid this kind of comparison.

4.2.4. Hybrid Models

Jiang and Conrath

Jiang and Conrath combined the edge-based approach and information content based approach.

As *link strength* $LS(c_i, p)$ they use a negative logarithm of the conditional probability of encountering an instance of the child concept c_i given an instance of its parent concept p. Substituting $IC(c)$ with $-log(p(c))$ we obtain:

$$LS(c_i, p) = -log\left(\frac{P(c_i \bigcap p)}{P(p)}\right) = -log\left(\frac{P(c_i)}{P(p)}\right) = IC(c_i) - IC(p). \tag{4.7}$$

The *edge weight* is derived from *link strength* and other factors such as node depth density around the node, the number of children the node has and the link type.

The simplification in Eq. (4.7) – replacing $P(c_i \bigcap p)$ with $P(c_i)$ is possible due to the following observation made by Jiang and Conrath [JC97]: "In the case of the hierarchical structure, where a concept in the hierarchy subsumes those lower in the hierarchy, this implies that $P(c)$ is monotonic as one moves up the hierarchy."

This assumption is reflected in the way information content is computed. As an occurrence of a concept c, not only the occurrences of the word representing the concept but also of terms that were direct or indirect hyponyms of this noun in WordNet are counted. In contrast to the approach used by Resnik, Jiang and Conrath used for their experiments a semantically tagged corpus SemCor, which has words annotated with WordNet synsets (refer to Subs. 6.1.6).

The distance between two nodes is a sum of edge weights along the shortest path connecting the two nodes. If we use link strength given by Eq. (4.7) as edge weight wt, then we get:

$$dist_{JC}(w_1, w_2) = \sum_{c \in \{path(c_1,c_2) - lso(c_1,c_2)\}} wt(c, parent(c)) = IC(c_1) + IC(c_2) - 2IC(lso(c_1, c_2)). \tag{4.8}$$

The *path* is a set of all nodes on the path. Since words w_1, w_2 may have multiple senses assigned, $c_1 = sen(w_1)$ and $c_2 = sen(w_2)$, where $sen(w)$ denotes the set of possible senses for word w. The function lso returns the lowest common subsumer (called lowest superordinate by [JC97]) of its arguments as proposed by Resnik. The sum in Eq. (4.8) goes over all nodes on the path except for lso.

Jiang&Conrath is a distance measure but can be transformed [PS08] to a similarity metric:

$$sim_{JC}(c_1, c_2) = 1 - \frac{IC(c_1) + IC(c_2) - 2IC(lso(c_1, c_2))}{2}. \tag{4.9}$$

Lin Similarity Measure

This measure has been proposed by Lin [Lin98]. The measure is grounded in the Similarity Theorem [Lin98]: "The similarity between A and B is measured by the ratio between the amount of information needed to state the commonality of A and B and the information needed to fully describe what A and B are." Symbolically, Lin gives the Eq. (4.10):

$$sim(A, B) = \frac{log\, p(\text{common}(A, B))}{log\, p(\text{description}(A, B))}. \tag{4.10}$$

For taxonomies such as WordNet, Lin concretized the formula by Eq. (4.11):

$$sim_{lin}(c_1, c_2) = \frac{2\, log\, p(lso(c_1, c_2))}{log\, p(c_1) + log\, p(c_2)}. \tag{4.11}$$

The function lso is defined in the same way as proposed by Resnik. The measure proposed by Lin is generic and can be used also in non-taxonomical contexts.

Pirro and Seco

Pirro and Seco suggested a hybrid measure coming out of the feature-based theory of similarity posed by Tversky [Tve77]. The details can be found in paper [PS08], for the purpose of this dissertation, it is sufficient to show the final formula:

$$sim_{P\&S}(c_1, c_2) = \begin{cases} 3IC(lso(c_1, c_2)) - IC(c_1) - IC(c_2) & \text{if } c_1 \neq c_2, \\ 1 & \text{if } c_1 = c_2. \end{cases} \tag{4.12}$$

Note that the Pirro and Seco similarity metric (in our opinion rather artificially) corrects

the problem with the Resnik similarity measure not giving similarity of 1 when computing the similarity between two identical concepts. Perhaps the most interesting aspect of the Pirro and Seco similarity metric is the fact that they compute IC from WordNet rather than from a standalone corpus. This *Intrinsic* Information Content is computed with the following formula:

$$IC(c) = 1 - \frac{log(hypo(c) + 1)}{log(max_{wn})}, \tag{4.13}$$

where function *hypo* returns the number of hyponyms of a given concept c and the constant max_{wn} expresses the total number of concepts in the WordNet noun taxonomy.

Computing IC values from WordNet addresses the problems experienced when the values are based on frequencies derived from a large unannotated corpus as in the Resnik experiments. In that case, all word occurrences are counted towards the probability of the given concept even if they are semantically unrelated given the context. While the Jiang and Conrath approach alleviates this problem by involving a semantically tagged corpora, it runs into the cost of producing a large enough semantically annotated corpus.

4.3. Implementations

There are various implementations of WordNet Similarity measures described in Sec. 4.2. Both systems proposed allow to use the JWordnetSim and JWSL libraries that are described in this section. There are also (relatively few) other libraries that implement WordNet similarity measures, such as the *Java WordNet::Similarity* developed by David Hope, which we do not describe here.

4.3.1. JWordnetSim

The de facto standard library used for WordNet similarity computations is the Ted Pedersen's *WordNet::Similarity*. Since the coding language used to implement software for this dissertation is Java, it is more convenient to use Java WordNet Similarity (JWordnetSim) library,[5] which was implemented by Mark A. Greenwood from the University of Sheffield. This convenience comes at a cost of support of a smaller number of measures. Namely, the library JWordnetSim covers Jiang& Conrath [JC97] and Lin Measure [Lin98]. JWordnetSim is implemented on top of the Java WordNet Library (JWNL).[6]

In the dissertation, we use this library in conjunction with WordNet 2.0. The library relies on `infocontent` files containing precomputed values of information content available along with the Ted Pedersen's *WordNet::Similarity* library. There are multiple variants of `infocontent` files available depending on the corpus used (British National Corpus, Brown corpus, Penn Treebank, SemCor, Shakespeare plays) and the computation method – refer to the next paragraph. Stopword list containing 199 words was used. For the SemCor corpus, two variants are available: with and without disambiguation (refer to Subs. 4.2.4).

IC Computation

The available computation methods are *default* and *Resnik* counting. In the default method, if a word appears in the text, the score of each WordNet concept associated with the word is incremented by 1. In the Resnik method, the increment is inversely proportional to the number of concepts associated with the word, with the maximum being 1. For example, if there are four concepts associated with a given word, the score of each of these concepts is incremented by 0.25. The method is selected by choosing appropriate IC file.

Dealing with Multiple Synsets

It is often the case that a word matches multiple synsets. The library offers two ways to deal with this situation. There is the `getSimilarity(Synset s1, Synset s2)` method, which computes similarity between specific synsets. This allows to apply the Most Frequent Sense (MFS) assumption by preselecting the first sense for the word.

The second option is the `getSimilarity(String w1, String w2)` method which returns the maximum similarity between the similarity maximizing combination of senses. We call this Synset Similarity Maximization (SSM).

[5] `http://nlp.shef.ac.uk/result/software.html` [Retrieved on 11 June 2012]
[6] sourceforge.net/projects/jwordnet/

Example 4.1 (WordNet similarity computation example – MFS). Computing Lin similarity between c_1 "football" and c_2 "football_player".

First, using the MFS assumption, the library takes the first sense for both entities *football.1* (the game) and *football_player.1* (athlete). The lowest common subsumer of these terms is the top-level concept *entity*.

The IC values are computed with Eq. (4.5) and Eq. (4.6).

$$IC(entity) = -log\left(\frac{5.1472464829244 97E7}{5.1472464829244 97E7}\right) = 0$$

$$IC(football\_player) = -log\left(\frac{15131.818253968253}{5.1472464829244 97E7}\right) = 8.132$$

Since "entity" has the probability of occurrence 1, the associated information content value is 0, and the resulting similarity is 0.

Although this may seem odd, this result is in line with the definition of semantic similarity as given in Sec. 4.2.

Example 4.2 (WordNet similarity computation example – SSM). Continuation of Example 4.1 with the SSM strategy. A WordNet lookup shows that there are two senses available from c_1 and just one sense for c_2. We can therefore try also computing the similarity between *football.2* and *football_player.1* and then choose the most similar combination of senses.

The lowest common subsumer for *football.2* and *football_player.1* is synset for "physical object".

$$IC(football.1) = -log\left(\frac{7910.5}{5.1472464829244 97E7}\right) = 8.780$$

$$IC(football.2) = -log\left(\frac{2854.5}{5.1472464829244 97E7}\right) = 9.800$$

$$IC(object) = -log\left(\frac{14068537.1104029565}{5.1472464829244 97E7}\right) = 1.2971$$

Using these information content values, the similarity can be computed:

$$sim_{lin}(c_1, c_2) = \frac{2\,log\,p(lso(c_1, c_2))}{log\,p(c_1) + log\,p(c_2)} = \frac{2 \times 1.2971}{9.800 + 8.132} = 0.144$$

Note that the number of occurrences of the concept in corpus (e.g. 7910.5 is the number of occurrences of football.1) are not integers. This is due to the fact that Resnik counting was used. The infocontent file used was `ic-bnc-resnik-add1.dat` for WNet 2.0 (British National Corpus, add1 smoothing, Resnik counting). The number of occurrences for entity relates to the noun part of speech.

Additional example using the JWSL library is present in Sec. 4.3.3.

Corpus Name	missing	covered	Standard Deviation
British National Corpus	19 499	73 699	464 521
Brown corpus	49 871	43 327	4 800
Penn Treebank	55 374	37 824	903
SemCor	62 017	31 181	928
SemCor Raw	63 396	29 802	231

Table 4.1.: Statistics relating to various sources used for information content computation. Source: computed from WordNet-InfoContent-2.0 files, obtained from `http://www.d.umn.edu/~tpederse/Data/WordNet-InfoContent-2.0.tar.gz`

Smoothing

If smoothing is performed, each concept starts with value 1. This ensures that even if there are no matching words, the information content for the concept is still nonzero.

Table 4.1 provides some means of comparison between the corpora. For each corpus, we have used the file with default computation method and without smoothing. It is clear that the British National Corpus (BNC) has the highest coverage with only 19,499 out of 93,198 synsets left uncovered. However, the standard deviation of the infocontent values for BNC is two orders of magnitude higher than of the second largest Brown corpus. Interestingly, if the synsets are ordered in a descending way, the first differences appear very close to the top of the list. Comparing the British National Corpus (BNC), the largest one, with the Semcor corpus, the only disambiguated one, the differences appear already on the 4th position. The first common three synsets are *entity*, *object* and *abstraction*. The 4th one for BNC is the verb *act* (synset 02296591), while for Semcor synset for *living thing*.

4.3.2. Performance

The JWordnetSim library is optimized for high performance computations. The library features caching, i.e. if the same similarity value is requested twice, the second request is returned a cached value.

The library is also accompanied by a utility generating a memory backed map from WordNet, which can then be used instead of a file-based storage. The memory map requires about eleven seconds to load (Intel i3 CPU, 7200 RPM disk drive), but the queries are then satisfied in significantly shorter time. Using the same configuration, the file backed representation takes approximately 0.009 seconds to compute similarity value in contrast to approximately 0.001 seconds for the map based representation.

Time required to retrieve a cached similarity is less than 0.001 second.

4.3.3. JWSL

The Java WordNet Similarity Library (JWSL) was developed by Pirro and Seco [PS08]. The main difference between this and other WordNet libraries is the way information content values are computed. They do not use corpora to compute information content measures but derive them directly from WordNet. This tackles the problem of sparsity highlighted in the previous Subs. 4.3.1, since in most corpora more than half of the WordNet synsets is left uncovered.

Dealing with Multiple Synsets

If multiple synsets match the input string, this library supports the SSM strategy using the `getSimilarity` function. It is also possible to specify a specific sense using function `getSenseSimilarity`.

Example 4.3 (Dealing with multiple senses in WordNet). Computing Lin similarity between "football" and "football_player". JWSL creates the following queries from these terms: "football.*" AND "football_player.*" and issues them against its Lucene-based index, retrieving $n = 2$ hits for "football" (football.1 and football.2) and $m = 1$ hit for "football_player". The synset football.1 denotes football the game, while football.2 the ball used in American football. The library then performs $n * m = 2$ Lin similarity computations.

The same formulas apply as in Example 4.1 with the difference that the IC values do not come from corpus statistics. The library selects the similarity maximizing combination of senses "football.2" and "football_player.1" with similarity 0.097.

Data Access

Technically, the library exploits a Lucene index to keep all the information content values, WordNet is not required. The disadvantage of this piece of software is the licensing policy. The library is not freely available and is provided by the authors upon written request.[7]

4.4. Use of WordNet in BOA and SCM

WordNet similarity measures are a basis of the SCM algorithm, which supports both JWordnetSim and JWSL implementations.

Most WordNet measures presented in this chapter were evaluated on the Czech Traveler dataset and the WordSim353 dataset. For details refer to Sec. 6.3 and 6.2 respectively, here we summarize the key findings:

- The Synset Similarity Maximization strategy produces consistently better results that the Most Frequent Sense strategy.
- What concerns the performance of individual measures, the best performing ones on both datasets were Resnik and Lin.
- A combination of multiple measures produces a better result than any single measure.
- The differences between geometric and arithmetic average as an aggregator are negligible.
- The differences in performance of the JWordnetSim and JWSL implementations are inconclusive, slightly better results are provided by the JWordnetSim library.
- On the WordSim353 dataset, the performance is poor in comparison with Wikipedia-based WSC algorithms, including BOA.

[7] The version provided by the Giuseppe Pirro did contain the source code, which was critical since the code had to be updated to Lucene 3 API in order to be able to work with the rest of the application. However, the executables not the source code for recreating the index were provided.

- On the Czech Traveler dataset, WordNet similarity measures surpass the BOA algorithm by a large margin.
- The choice of the `infocontent` file has negligible impact.

The results on the standard WordSim353 dataset do not stand comparison with Wikipedia-based WSC algorithms. The BOA algorithm is a Wikipedia-based WSC algorithm, but it allows to use WordNet similarity measure as a term-weighting function, as a positive term list and a lemmatizer. Experiments presented in Sec. 6.4 indicate that the use of the WordNet-based term-weighting function improves results on the WordSim353 dataset. Also, the use of WordNet as a positive term list and lemmatizer has a positive impact on the results. With the help of WordNet, the performance of the BOA algorithm gets closer to the state-of-the-art ESA algorithm.

Surprisingly, the SCM algorithm (i.e. the existing WordNet similarity measures) beats the BOA algorithm on the Czech Traveler dataset. Our conjecture is that this may be related with the different type of the tasks handled. In WordSim353, a similarity of two pairs of words is compared. These two words typically fall into a similar category – both are well known abstract words, and are mapped to comprehensive Wikipedia pages with good link neighborhood.

In contrast, on the Czech Traveler dataset we compute a similarity between an abstract concept, which is represented by several named entities, and a specific noun phrase in image caption, often also a named entity. The named entities involved are mapped generally to shorter Wikipedia pages with less rich neighborhood than entities in the WordSim353 dataset. While BOA method relies on the availability of enough discriminatory textual content in the entity article and the associated modalities, the two best performing WordNet similarity measures are Resnik and Lin.

Lin measure requires the information content of the two entities and of their lowest common subsumer. Resnik measure does not even require the former. These values are available for all the entities. To derive the lowest common subsumer, only the hypernym relationship is required. The performance of WordNet measures therefore does not depend on the "length of the entity description", once the entity is WordNet, all entities are "equal". Another interrelated factor is that SCM uses THD and head noun extraction to map named entities to WordNet, this is required since the coverage of WordNet is limited. In contrast, BOA is able to map the entities directly to Wikipedia. In the context of the rather general classes in the Czech Traveler dataset this can be viewed as an disadvantage. For example, entity "Buje" is mapped to WordNet concept "town", while BOA uses the relatively short Wikipedia article "Buje" to build the classifier.

5. Wikipedia as a Knowledge Source

Wikipedia is a central resource for this dissertation. It is used as a corpus for rule based NLP in the THD method within the SCM classifier and it is used for statistical NLP in the BOA classifier. There is a growing amount of work dealing with extraction of various types of knowledge from Wikipedia. Consider for example the *Workshop on Wikipedia and Artificial Intelligence* which was held in 2008 and 2009 in conjunction with the AAAI conference. Among the organizers of this workshop are NLP researchers including Evgeniy Gabrilovich, one of the authors of ESA [GM07], the best performing word similarity algorithm to date.

In this chapter we focus only on the following two narrow areas: (1) targeted hypernym discovery from Wikipedia entity pages, (2) mapping noun phrases to Wikipedia entity pages. While area (1) is relevant for SCM, area (2) is important for both SCM and BOA. In contrast to the previous two chapters, which were mainly summarizing existing work, the main method in this chapter is experimental exploration of the selected topics. The reason for this is that relevant work is not available.

This chapter is organized as follows. Sec. 5.1 examines the potential of using Wikipedia for THD and motivates three experiments that should underpin the use of Wikipedia as a knowledge source in the SCM and BOA algorithms. We begin with two experiments evaluating the influence of article popularity on THD performance. While the experiment in Sec. 5.2 measures the popularity by links, the experiment in Sec. 5.3 measures the popularity by hits. Sec. 5.4 takes a more qualitative look on THD performance by using a real-world dataset and presenting a detailed analysis of the individual extraction results. This third experiment also addresses the problem of mapping noun phrases to entity pages. Sec. 5.5 gives a summary of experimental observations.

5.1. Motivation and Experimental Setup

WordNet is usually considered to be a gold-standard dataset for training and testing hypernym discovery algorithms [SJN05]. Its structured nature and general coverage make it a good choice for general disambiguation tasks. However, WordNet is less useful for dealing with named entities; while some key named entities (names of countries, U.S. presidents) are covered, it is out of the scope of the WordNet project to provide complete or even consistent named entity coverage.

Various free-text corpora have been used to overcome the problem of WordNet sparsity. State-of-the-art approaches based on mining patterns from text already achieve a higher F-measure than WordNet, when human judgment is used as the ground truth. Interestingly, Wikipedia as a free and comprehensive source of information plays a vital role in these efforts; one of the best results [SJN05] was achieved by extending the TREC corpus with articles from Wikipedia.

In the scope of the targeted hypernym discovery, Kazama and Torisawa [KT07] were probably first to use THD-like algorithm on Wikipedia encyclopedia to improve the accuracy of

a NER system. The authors noticed that Wikipedia contains many articles defining named entities. For a given named entity extracted from text, the algorithm of [KT07] automatically found a Wikipedia entry for this entity and applied lexico-syntactic patterns to extract a hypernym from the first sentence of the article. TagChunk [DM05] was used to tag parts of speech and noun phrases in the article text.

Our algorithm, which is in detail described in Sec. 1.3, performs nearly identical handling of articles and their text and it relies basically on a similar set of assumptions as implicitly imposed by [KT07]:

1. Wikipedia contains articles (*entity pages*[1]) on many commonly appearing entities,
2. entity pages tend to contain a hypernym for the entity described in the article,
3. a limited number of lexico-syntactic patterns can be used to extract the hypernym from most articles,
4. the first match of the pattern in the article, or within its processed part (first sentence, first paragraph, first section), is generally the best.

For the individual assumptions we can find certain grounding also in other algorithms that mine Wikipedia article texts. For example [SP06] processes only the first paragraph of the Wikipedia article. Paper [Cuc07] notes that "most articles in Wikipedia are associated to an entity/concept". In experiments conducted in this chapter We provide results that underpin some of these points.

5.1.1. Wikipedia Manual of Style

Upon a manual inspection of hypernym discovery results on the Czech Traveler dataset we observed, with a few exceptions, that the first match of an ideal Hearst pattern (refer to Subs 3.2) in the article provided an informative and specific hypernym. Indeed, we found the vast majority of Wikipedia articles to open with clear definitions following the pattern "XYZ is a ... *close hypernym*." Exceptions included cases such as "XYZ is a *cross* between a A and B". In this case, the word *cross* matches the lexico-syntactic pattern "XYZ is a ?" but cannot be accepted as a useful hypernym for XYZ.

We consider this rigidity in opening sentences surprising. Such a clearness and uniformity of articulation could be expected from an expert-created encyclopedia or thesaurus but not from a resource collaboratively created by unpaid volunteers whose only training for the task comes, in general, from reading the Wikipedia guidelines. The Wikipedia's *Manual of Style*[2] has, indeed, a special section on first sentences, which instructs authors to

> "put the article title as the subject of the first sentence".

A special article on the lead (introductory) section[3] states that:

> [first paragraph] "needs to unambiguously define the topic for the reader".

[1] This term is introduced in [Cuc07] to denote "an article that contains information focused on one single entity, such as a person, a place, or a work of art".

[2] `http://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style#First_sentences` [Retrieved on 11 June 2012]

[3] `http://en.wikipedia.org/wiki/Wikipedia:Lead_section` [Retrieved on 11 June 2012]

hypernym	frequency	# distinct	hypernym	frequency	# distinct
country	2598	152	term	369	78
city	1436	284	form	344	40
name	1270	281	town	287	97
player	578	250	cricketer	276	97
day	564	131	adjective	260	6
month	554	15	golfer	229	88
club	537	167	world	221	24
surname	515	185	team	220	52
capital	454	79	organization	214	38
state	416	60	second	212	1

Table 5.1.: Top 20 extracted hypernyms from Wikipedia for entities appearing in CONLL'03 dataset in [KT07]

Remarkably, Wikipedia contributors seem to follow these guidelines and even exceed them by refraining from the use of a more varied vocabulary when writing the opening definitions. For example, instead of

"Diego Armando Maradona is a former Argentine football *player*",

which is an opening sentence of Wikipedia article on Maradona as of the time of writing, the article title could start e.g. by

"Diego Armando Maradona, a former Argentine football player, played in four ...",

or even worse

"D.A. Maradona was a *backbone* of Argentine football...".

The list of top 20 most frequently extracted words by Hearst-like lexico-syntactic patterns reported in [KT07] is present in Table 5.1. The authors used word sequences extracted from CONLL 2003 dataset and tried to map them to Wikipedia articles. If the mapping succeeded, the word sequence was considered in our terminology an entity, and a category label was extracted using a Hearst pattern ("is", "was", "are", "were"). The last noun in the matching noun phrase was used as the category and is presented in Table 5.1.

5.1.2. Experiment Setup

This subsection describes three experiments that were conducted within the scope of this dissertation on Wikipedia.

Experiment 1 and 2

Experiment 1 and Experiment 2 explore if there is a correlation between the popularity of a Wikipedia article and the successfulness of hypernym discovery from this article. The popularity of Wikipedia articles can be measured in several ways. Experiment 1 takes the viewpoint of Wikipedia contributors and uses the number of inbound links from other Wikipedia articles. In turn, Experiment 2 takes the viewpoint of Wikipedia readers and uses the number of hits the article receives as the measure of popularity.

As a hypernym discovery algorithm we used our THD implementation described in Chapter 1. It should be noted that these choices were influenced by our previous work [CKN+08, KCN+08a] (the latter presented in Chapter 1), which evaluated the usefulness of hypernyms discovered from Wikipedia for image retrieval. These two experiments were published in our paper [KCN+08b].

Experiment 3

When we consider applying Wikipedia-based THD or a BOA classifier on a dataset, it is useful to have some notion of the number of entities appearing in the dataset that we can expect to have an entity page in Wikipedia. For Experiment 3, we extract entities from a real-world dataset, and identify a subset of entities, which cannot be mapped to WordNet. Using this "hard" dataset, two evaluations are made. First, entities are mapped to Wikipedia articles and the correctness of the mapping is evaluated. Second, our THD implementation is used to extract a hypernym for entities, where the mapping did not fail, at two time points – in 2008 and 2011. The THD result is then manually assessed.

5.2. Experiment 1: Influence of Article Popularity (Links)

This experiment aimed to evaluate the influence of article popularity as measured by the number of inbound links from Wikipedia articles on the performance of THD. The underlying rationale is that the higher the number of linking articles, the higher the chance that other contributors will intervene if an article does not comply with the guidelines or its opening section is poorly/unclearly written.

Wikipedia MediaWiki Lucene Search Extension can use article popularity as measured by the number of articles that link to it as one of the ranking factors in addition to text-based relevance.[4] However, since we are only interested in article popularity, we try to mitigate the influence of text-based relevance by only involving articles whose title contains the entity for which the hypernym is sought, assuming that the part of relevance coming from the textual similarity between the article and the query is the same for all the retrieved articles (up to a certain relevancy threshold). Manual inspection of the results showed that this technique was effective and the relevance measure generally reflected the relative popularity of the article subject in our dataset.

5.2.1. Dataset and Ground-truth

As test hypernym queries we used the surnames of ten top-rated NHL goalkeepers: Nabokov, Brodeur, Lundqvist, Luongo, Leclaire, Giguere, Miller, Bryzgalov, Turco and Kiprusoff. We already used this test set in our earlier work [CKN+08], where we found these words to provide enough ambiguity, as each represents a surname of several important persons from different fields, in addition to other meanings such as names of jobs, places or companies.

For each query, the first section of all returned articles from Wikipedia up to a relevancy threshold of 50% was downloaded. Redirects were followed but disambiguation articles and articles where the query term was not in the title were discarded. The resulting collection contained 131 documents (articles).

[4] `http://www.mediawiki.org/wiki/Extension:Lucene-search` [Retrieved on 11 June 2012]

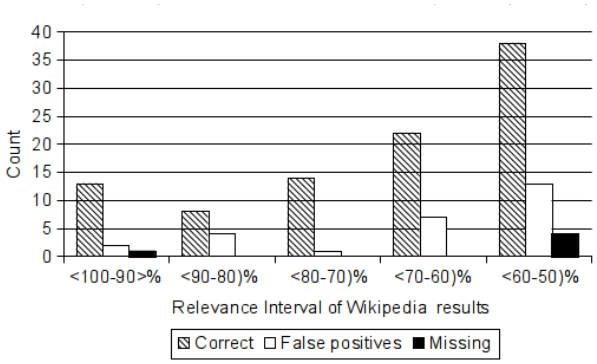

Figure 5.1.: Impact of article popularity on THD performance

Two human annotators annotated each of the documents. They were instructed to only mark the first hypernym per document (as does the used THD algorithm) with respect to the query, regardless of how much more general this hypernym would be than the query. The annotation was not restricted to the context of ice hockey; hypernyms expressing all conceivable meanings of the original query were considered. The annotators agreed on 96% (126) of annotations, which formed the ground truth.

Results

The algorithm discovered 95 out of 126 hypernyms on which annotators agreed. Fig. 5.1 shows the distribution of the THD outcome depending on article popularity. The evaluation approach follows the methodology taken by the GATE Annotation Diff tool [CMB+12]. If the word marked as hypernym by the algorithm matched the ground truth then it was considered as *correct*, an incorrect hypernym returned by the algorithm was counted as *false positive* error, *missing* error was counted when the system did not discover an annotated hypernym.

We performed a statistical test to explore the significance of association between the article popularity (given by the respective relevance interval) and the successfulness of THD (1 for correct hypernym discovered, 0 otherwise). The test used was one-sided Kendall's Tau B [Mut99], which makes adjustment for ties and is also suitable for binary variables. A value of zero indicates the absence of association, while -1 or 1 mark perfect negative/positive association. Since the one-tailed Kendall's Tau B is equal to 0.1281 (p-value of 0.055), we cannot reject the null hypothesis at a 5% significance level that there is no correspondence between article popularity (based on links) and the successfulness of hypernym discovery.

It should be noted that the experimental results may be affected by the narrow character of the dataset and by the residual influence of article-query relevance. It may be worthwhile to repeat the experiment on a larger dataset and use the number of in-links directly.

5.3. Experiment 2: Influence of Article Popularity (Hits)

Another way of measuring the popularity of an article is the number of hits (views) it receives from the general public. Again, since anyone can edit Wikipedia articles,[5] the higher the

[5] abstracting away from the locked articles

number of hits, the higher the chance that a random user would not only contribute with new content or fix a factual error, but also intervene if the article would violate the Wikipedia quality guidelines, the adherence to which is an important prerequisite for hypernym discovery.

5.3.1. Dataset and Ground-truth

This experiment was carried out with the sample of 100 articles describing named entities, which were randomly selected using the Wikipedia's random article link. The ground truth was established in a similar manner as in Experiment 1, but hypernyms were extracted for the topic of the article and not for a query. Additionally, articles were only annotated by one annotator.[6] Article titles were used as queries for hypernym and the articles as the corpus. Out of the 100 articles, 98 contained at least one correct hypernym.

For comparison regarding the size of the sample, study [SJN05] randomly selects 5,387 noun pairs from a free-text corpus out of which 5,122 noun pairs were annotated as unrelated and the rest was split among 131 coordinate and 134 hypernym pairs with named entities accounting for more than 60% of the labeled noun-hypernym pairs.

5.3.2. Results

The system failed to extract the correct hypernym from 14 articles:

- in 8 articles a Hearst-like pattern was not present,[7]
- in 6 articles a Hearst-like pattern was present but not matched by the grammar.

A detailed analysis of the results depending on entity type is depicted on Fig. 5.2. An encouraging result from the point of view of integration of THD with WordNet-based classifiers like SCM is that all the discovered hypernyms were mappable to WordNet with a disambiguation accuracy of 87% for the most frequent sense synset.

We evaluated whether the inability of the system to extract a hypernym is dependent on the number of hits each of the 100 articles obtained during a one-month period.[8] The range of hits was between 1 (for *Kielpino Kartuskie*) and 25.253 (for *Dead Space (video game)*), with the median value being 237. The result of extraction was marked as either 1 (success) or 0 (all other cases). The different kinds of error were alone too rare to be tested separately.

The test used was the same as in Experiment 1 – Kendall's Tau B. The value of the Kendall's Tau B in Experiment 2 was -0.037 (p-value of 0.320). This result hints that there is not a statistically significant correspondence between article popularity (based on hits) and the successfulness of hypernym discovery.

[6] Exp. 1 showed that annotations by two humans do not significantly differ.

[7] Interestingly, the hypernym was a part of the article name in 6 cases, in 1 case there was no hypernym. The last case has interesting history. In the time between the original experiment and its verification motivated by the preparation of the camera ready version of the paper [KCN+08b], Wikipedia contributors corrected the first sentence of this entry on R. E. Holz from "Richard E Holz, ... an American brass band composer,.." to then extractable "Richard E Holz was an American brass band composer...". At the time of writing this dissertation, the first sentence reads "Richard E Holz (30 October 1914 – August 1986), was an American brass band composer,..." which still contains an extractable hypernym.

[8] During May 2008, using the `http://stats.grok.se/en` tool.

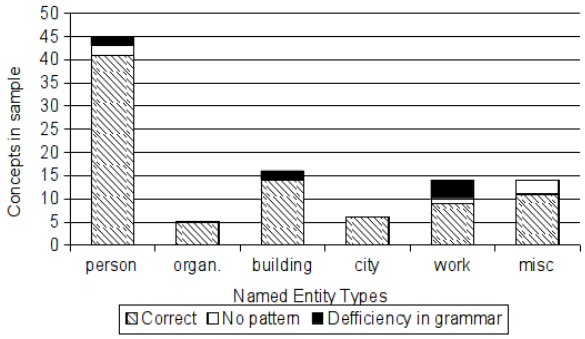

Figure 5.2.: THD accuracy per named entity type, ("work" refers to work of art or a literary work)

5.4. Experiment 3: Mapping and THD on Real-World Data

In this experiment, THD is used to resolve the entities from the Czech Traveler dataset for which neither the entity nor its head noun can be found in WordNet. We focus on these ("hard") entities, since matching WordNet concepts with Wikipedia articles was already studied (refer e.g. to [FS10, SKW07]). It should be noted that the work presented here is an updated and reworked version of an experiment we published in [KCN+08a].

5.4.1. Dataset, Ground-truth

The Czech Traveler dataset described in Subs. 6.1.3 was used. This dataset contains 186 entities. We used all 41 unique entities for which not even the head noun could be mapped to WordNet.

Example 5.1 (Selection of entities from Czech Traveler dataset). The entity "Crimean Tatar's camel" from the Czech Traveler dataset was not included, because "camel" can be found in WordNet. On the other hand, "Karst Cave Villenica" was included since "Villenica" is not in WordNet.

A human evaluator used Wikipedia[9] to find the article describing the entity (entity page) and from this article extracted the first one-word hypernym for the entity. The SCM/THD algorithm works virtually in the same way, therefore it is natural to evaluate if the entity pages and consequently the hypernyms extracted by the machine and a human evaluator match.

It should be noted that this is a more stricter criterion than used in the SCM Experiment presented in [KCN+08a], where a hypernym is considered correct if a human annotator judges it to fall to a more broadly defined class (such as vegetation, structure). When evaluating entity presence, we also do not demand that the hypernym is mappable to WordNet. This is evaluated in Subs. 5.4.4.

[9]Live English Wikipedia on September 8, 2011, the evaluator was the author of this dissertation

number of entities	41
mapped to entity page – THD	36
mapped to entity page – ground-truth	37
mapped to entity page by THD *and* ground-truth	33
mapped to *same* entity page by THD and ground-truth	27
THD hypernym matches ground-truth hypernym (on results from previous row)	27

Table 5.2.: Results on a subset of 41 entities from the Czech Traveler dataset

Example 5.2 (Evaluating the correctness of extracted hypernyms). For entity "Albanian guide Kamil", the system extracted hypernym "statesman". This is marked as incorrect by the annotator for the purpose of this experiment.

Note that in the experiment performed in paper [KCN+08a] this would be considered as correct, since out of the available target classes the closest one is "organism".

5.4.2. Results

The results are summarized by Table 5.2. Out of the 41 entities, for 27 entities THD and human evaluator retrieved the same entity article. Interestingly, in all of these articles the hypernym was not only present and extracted by THD, but it also matched the ground-truth human-picked hypernym.

Detailed results per entity are given in Table 5.3. For interestingness sake, hypernyms retrieved in 2008 for our paper [KCN+08a] (if available) are also included.

5.4.3. Sources of Error

This subsection discusses several interesting mappings (mostly failures to map) for entities listed in bold in Table 5.3.

Albanian guide Kamil The LHSHearstBody macro does not cover the CC POS tag (conjunction, coordination), which appears in

Kamil is a Polish, Czech, and Slovak given name

Additionally, the correctness of the ground-truth entry "Kamil" (the name) for noun phrase ("Albanian guide Kamil") is dubious.

Apolonia The system picked the first sense/hit, while the correct one was second.

Chersonesus The system picked a hypernym from a disambiguation page:

Chersonesos or Chersonesus is the Greek for "peninsula". The term appears in various ancient toponyms: .. Chersonesos (Lyctus), an ancient Greek city on Crete that was the harbor of Lyctus ...

Table 5.3.: Entities not mappable to WordNet

The list of unique entities from the Czech Traveler dataset not mappable to WordNet (even by head noun) for which THD results from 2008 are available. We present THD/SCM results (**Retrieved** columns) along with the manual hypernym discovery results (**Groundtruth** columns), both from 8th September 2011.

Column **Pos**ition (Pos) gives the position of a Wikipedia page describing the entity (if in top 20 search results). For multi-word entities, letter F (for full) denotes that entire noun phrase was used for the search, while letter H (for head) indicates that the preceding option failed and the result was produced by using the head noun only. Value 0 indicates that neither of them produced a successful result. Column **Page** lists the title of the entity page and column **Hypernym** the hypernym discovered.

The **H**ypernym **2008** column (H 2008) lists hypernym retrieved for the 2008 paper [KCN+08a]. Value '???' indicates that a hypernym was retrieved but was not accidentally stored. Column **P**age **C**orrect (PC) gives 1 if THD/SCM extracted hypernym from the page annotated as correct, 0 if it was extracted from a different page and empty if no ground-truth is given. The column **H**ypernym **C**orrect (HC) is empty if PC$\neq$ 1, otherwise it gives 1 if the extracted hypernym matches the ground-truth and 0 stands for no match. The lines in bold are detailed from the THD perspective in Subs. 5.4.3. The lines in italics are detailed from the WordNet mapping perspective in Subs. 5.4.4.

Entity		Retrieved (2011)			Groundtruth (2011)			(2011)		(2008)
ID	**Noun phrase**	**Pos**	**Page**	**Hypernym**	**Pos**	**Page**	**Hypernym**	**PC**	**HC**	**H 2008**
1	**Albanian guide Kamil**	**2H**	**Kâmil Pasha**	**statesman**	**1H**	**Kamil**	**name**	**0**		**sultan**
2	**Apolonia**	**1**	**Apolonia Fier**	**club**	**2**	**Sozopol**	**town**	**0**		
3	Arefu	1	Arefu	commune	1	Arefu	commune	1	1	commune
4	Bakhchisarai	1	Bakhchisarai	town	1	Bakhchisarai	town	1	1	
5	Berat	1	Berat	town	1	Berat	town	1	1	
6	Buje	1	Buje	town	1	Buje	town	1	1	town
7	*Calanques de Piana*	*1*	*Calanques de Piana*	*calanques*	*1*	*Calanques de Piana*	*calanques*	*1*	*1*	*???*
8	**Chersonesus**	**1**	**Chersonese (disambiguation)**	**Harbor**	**1**	**Chersonesos Taurica**	**city**	**0**		
9	Chufut-Kale	1			1	Chufut-Kale	fortress	0		
10	Curtea de Arges	1	Curtea de Arges	city	1	Curtea de Arges	city	1	1	striker
11	delphinarium	1	Dolphinarium	aquarium	1	Dolphinarium	aquarium	1	1	aquarium
12	**Dhermi**	1	**Hypapante Church, Dhërmi**	**church**	**1**	**Dhermi**	**village**	**0**		**lake**
13	Durau	1	Durău	resort	1	Durău	resort	1	1	resort
14	**Eski Kermen**	**1H**	**Kermen**	**town**						
15	Foros	1	Foros	town	1	Foros	town	1	1	member
16	Gjirokaster	1	Gjirokastër	city	1	Gjirokastër	city	1	1	city
17	GR20	1	GR 20	footpath	1	GR 20	footpath	1	1	footpath
18	Inkerman	1	Inkerman	town	1	Inkerman	town	1	1	town
19	**Istarske Toplice**				**1F**	**Istarske Toplice**	**resort**	0		**town**
20	Jablonica	1	Jablonica	village	1	Jablonica	village	1	1	village
21	Jezersko	1	Jezersko	municipality	1	Jezersko	municipality	1	1	village
22	Karst Cave Vilenica	1F	Vilenica Cave	cave	1F	Vilenica Cave	cave	1	1	cave
23	**Khersones**	**2**	**Khersones (ship)**	**ship**	**1**	**Chersonesus Taurica**	**colony**	**0**		**city**
24	Korce	1	Korçë	city	1	Korçë	city	1	1	City
25	Lacul Balea	1	Bâlea Lake	lake	1	Bâlea Lake	lake	1	1	lake
26	Les Cascades des Anglais	0	Anglais	rhythm						woodwind
27	Llogare	0								
28	Logarska Dolina	1	Logarska Dolina	settlement	1	Logarska Dolina	settlement	1	1	
29	*Massandra*	*1*	*Massandra*	*townlet*	*1*	*Massandra*	*townlet*	*1*	*1*	*???*
30	**Monte d oro**	**1F**	**Monte d oro**	**lord**	**1F**			**0**		**town**
31	Motovun	1	Motovun	village	1	Motovun	village	1	1	
32	Oprtalj	1	Oprtalj	community	1	Oprtalj	community	1	1	
33	Place St Nicolas	0						1	1	alliance
34	Qeparo	1	Qeparo	village	1	Qeparo	village	1	1	
35	Sarande	1	Sarandë	capital	1	Sarandë	capital	1	1	
36	Sevastopol's delphinarium	1H	Dolphinarium	aquarium	1H	Dolphinarium	aquarium	1	1	aquarium
37	Skanderbeg	1			1	Skanderbeg	lord	0		figure
38	**Syri i Kalter**	**1H**	**Alan Kalter**	**announcer**	**1F**	**Blue Eye, Albania**	**spring**	**0**		**announcer**
39	Umago	1	Umago	city	1	Umago	city	1	1	
40	Velika Planina	1F	Velika Planina	plateau	1F	Velika Planina	plateau	1	1	plateau
41	**Vlore**	**2**	**Vlorë County**	**shore lines**	**1**	**Vlorë**	**town**	**0**		**club**

Note that "Greek" was not matched as hypernym, because it was was marked as adjective by the POS tagger.

Dhermi The article contains two cardinal numbers (CD) between the verb and the hypernym, which is unaccounted for by the `LHSHearstBody` macro.

> Dhërmi (Greek: ..., Drymades) is one of the nine villages Istarske Toplice (Terme Istriane, o Bagni di Santo Stefano), a thermal health resort in the central part of Istria, Croatia ...

Eski Kermen A Wikipedia article for this entity (an underground town) does not exist. There is however a Wikipedia article on city "Kermen" yielding a hypernym "city", which can be considered as correct.

Istarske Toplice This article's opening sentence is unfavourably worded as the verb is missing.

> Istarske Toplice (Terme Istriane, o Bagni di Santo Stefano), a thermal health resort in the central part of Istria, Croatia.

Khersones For Khersones, a wrong (#2) sense (Khersones the ship) was picked, because the title of the page "Chersonesus Taurica" does not exhibit sufficient Jaro-Winkler string similarity with "Khersones".

Monte d'oro For Monte d'oro there is no Wikipedia page, the THD using head noun yielded an article for another entity.

Syri i Kalter There is a problem in the application handling the redirect from "Syri i Kalter" to article "Blue Eye, Albania", probably due to insufficient Jaro-Winkler similarity of the two strings. The hypernym from "Blue Eye, Albania" is extracted correctly.

Vlore failed because the article contains the word "of" tagged as IN between the verb and the hypernym. The IN tag (preposition or subordinating conjunction) is not covered by the `LHSHearstBody` macro (refer to Subs. 1.3.3). The system therefore resorted to #2 hit producing the incorrect result.

> Vlorë (known also by several alternative names) is one of the biggest towns ...

5.4.4. Mapping to WordNet

The extracted hypernyms map to WordNet very well, with all but two mapping directly. The problematic ones are calanques (entity Calanques de Piana) and townlet (entity Massandra) – marked with italics in Table 5.3.

Calanques de Piana THD on article "Calanques de Piana" yields correct hypernym "calanques". Unfortunately, this hypernym is not mappable to WordNet.

Assuming recursive hypernym resolution is allowed in the SCM/THD algorithm, the system issues a THD hypernym query for "calanques" which succeeds with word "inlet"; this word is already in WordNet.

Calanques de Piana (article)-> calanques (hypernym) -> Calanques (article)-> inlet (hypernym)

Massandra As a first hypernym candidate, THD yields the semantically correct hypernym townlet. Unfortunately, since townlet cannot be mapped to WordNet, the system proceeds to a less relevant article and extracts the hypernym "Asteroid".

5.4.5. Evolution of Extraction Results between 2008 and 2011

There are several straightforward observations:

- the number of hypernyms not retrieved dropped from 10 to 5,
- out of the 11 newly extracted hypernyms, 8 were correct (with Apolonia, Chersonesus and Eski Kermen[10] producing the two incorrect ones,
- hypernym for three entities can no longer be extracted (Skanderbeg, Istarske Toplice, Place St Nicolas).

While the size of the sample does not allow us to draw statistically sound conclusions, it is sufficient to illustrate that the continuous evolution of Wikipedia does indeed have a measurable positive impact on THD on a real life dataset.

There are of course shortcomings as well, the article "Skanderbeg" was originally phrased in simpler terms yielding hypernym "figure", which is as a matter of fact still present as of 8/9/2011 in the first paragraph, but it was rephrased to an unextractable form as discussed in Subs. 5.4.3.

This hints that a possible improvement in THD results could be achieved by using an older version of the article if hypernym discovery from current article fails, before resorting to analyzing an article lower on the result list. Table 5.3 shows that all three cases (Albanian guide Kamil, Khersones, Vlore) when a second hit was taken, resulted in choosing a wrong entity article.

5.5. Summary of Experimental Observations

In Experiments 1 and 2, we selected one of the factors that may influence the success of THD – the popularity of the article – and inspected its influence on the success of the extraction. We conjectured that for articles covering less popular topics the Wikipedia authoring guidelines are less rigidly applied, which may result in a worse performance of hypernym discovery algorithms. The preliminary experimental results carried out altogether on 231 Wikipedia

[10] Technically, the hypernym for Eski Kermen is correct, but it is marked incorrect as it is extracted from page for semantically different entity.

documents do not support this hypothesis. Nevertheless, it should be noted that the value of the test criterion in Experiment 1 was very close to the critical value for $\alpha = 5\%$. Since Experiment 1 was conducted on a sample from a specific domain and the article popularity was inferred from search relevance results, which might have introduced additional error, a larger scale experiment is indispensable to give the final answer.

In Experiment 3, THD was performed on a smaller number of entities than in the previous two experiments, but the reasons for failure were investigated in detail.The result of the experiment along with the analysis of errors can serve as a basis for improving the grammar in further work.

Using the intermediate results from the experiments, we can also roughly determine some probabilities for Wikipedia as an "entity resolution service":

Ratio of pages describing named entities to all articles In order to draw 100 entity pages in Experiment 1, it was necessary to draw 130 articles. This indicates that about 77% of Wikipedia articles are entity articles. The definition that we used to differentiate between entity and non-entity was very strict. As an entity we considered only a physical object. For example, "Politics of New Jersey", "1988 Calder Cup Playoffs", "List of United Kingdom locations: U.", "1963 New York Jets season" were not considered as entities.

Ratio of named entities in free text image captions to all entities The Czech Traveler dataset used in the Experiment 3 contains 101 named entities[11] (proper nouns), which accounts for around 54% of the total of 186 entities in the annotations. This gives a very rough estimate of the ratio of named entities to all entities in this kind of image annotations.

Coverage of entities The results indicate that if the dataset covers generally popular entities, such as hockey players in Experiment 2, the entity page virtually always exists. Experiment 3 considered only the "hard" entities with not even the head noun mappable to WordNet. These entities were almost exclusively[12] named entities. Human annotator was able to find an entity page for 36 entities out of 41.

5.5.1. Ideas for Future work

The odds that entity will appear in Wikipedia are clearly tied to some features of the entity such as its proliferation (well-known, almost unknown) or type (physical, abstract, ...).

If the dataset involves *prolific* people and places, many named entities from the dataset will be likely to occur in Wikipedia. It is common sense that the opposite is true: the less famous the entity, the smaller the chance of the existence of a Wikipedia article covering the entity. The same principle applies to the age (or *recency*) of the entity. If the entity drew public interest quite recently, it may not have a Wikipedia article.

An extreme case (though not that uncommon in practice) are datasets containing almost exclusively "private" entities with very low proliferation and no Wikipedia presence. Even if this is the case, THD may still achieve satisfactory results if used from a classification application such as SCM. If we consider person names, then mapping "Walter Miller" an ordinary person

[11] Note that the number of named entities was not mentioned in Subs. 5.4 but comes from the description of the dataset in Subs. 6.1.3.

[12] With two exceptions (delphinarium)- id 11 and id 36.

without a Wikipedia page to "Walter B. Miller" (the first sense in Wikipedia[13]) is indeed incorrect. However, if we extract hypernym "anthropologist", the proposed Wikipedia-based algorithms will still (hopefully) correctly label the entity as *person* if the task is to decide between the three or four ENAMEX *person*, *organization*, *location* classes, which is actually as far as the classical NER task goes.

Proliferation and recency may also influence the success of hypernym discovery provided the entity page exists. When working on papers [CKN+08, KCN+08a] we got the impression that hypernym discovery from shorter, less elaborate Wikipedia articles, which often describe uncommon entities, tends to be less successful than extraction from long articles on popular topics. For example, definitions of physical entities will be intuitively more likely to feature a simple Hearst pattern than of abstract entities. An investigation in this direction may be an interesting and practically important subject of future work.

[13] As of June 8, 2011

6. Evaluation

The purpose of this chapter is to quantitatively evaluate the two main approaches to entity classification proposed and implemented within this dissertation – SCM and BOA algorithms.

Sec. 6.1 reviews the possibilities for the choice of the experimental dataset for the entity classification task. Sec. 6.2 evaluates SCM on the WordSim353 dataset and Sec. 6.3 on the Czech Traveler dataset. The BOA algorithm is evaluated on the WordSim353 dataset in Sec. 6.4 and on the Czech Traveler dataset in Sec. 6.5. In Sec. 6.6 we provide the final assessment of the experimental results.

6.1. Datasets

The choice of the experimental dataset is of paramount importance. Unfortunately, there are not many options, in fact, the author is not aware of any publicly available dataset that would fit the entity classification problem. The requirements on the dataset follow from the statement of the thesis:

- moderately large and diverse set of target classes,
- input pieces of text have moderate length of several words up to two sentences,
- input pieces of text share the same *global* context.

In addition to these requirements, there are the usual:

- entities in the input text are assigned ground-truth created by multiple annotators,
- the dataset is freely available.

Concerning the minimum size of the dataset, the situation is not clear. All methods classify entities, therefore the number of entities is the principal measure of the dataset size. SCM and THD do not require training examples. BOA classifier also does not require any labeled data apart from the target classes being mapped to Wikipedia articles, however, it *allows* to add positive examples for each class. If the effect of the additional examples on the classification results is to be evaluated, some training data need to be available. In general, the size of the training data required to evaluate the proposed algorithms is much smaller than for fully supervised methods. These would require 10:1 or 5:1 ratio between training and testing data to estimate the performance possible with the current amount of data [Agi07].

There are multiple datasets that meet some, but none that meet all of these criteria. This section tries to give a representative account of the datasets (and types of datasets) available. In Subs. 6.1.3 a new dataset meeting all criteria, except for the free availability, is introduced.

6.1.1. Named Entity Recognition

There are established test collections for the related Named Entity Recognition tasks. These collections are in principal applicable, however most dataset interpret the NER task as classification into four general classes (PERSON, LOCATION, ORGANIZATION, MISCELLANEOUS). Performance of a THD/SCM or BOA implementation on such a general set of classes has little generalization potential for use with datasets involving many specific target classes.

For example, using hypernyms extracted from Wikipedia with lexico-syntactic patterns as features for a Conditional Random Field NER classifier on the CONLL 2003 dataset in [KT07], improved the F-Measure only by 3.03 points compared to the baseline, which does not use any gazetteer like knowledge. In reaction to this, the authors of the paper express their belief that the advantage of hypernyms unwinds only on a fined-grained task with many target classes.

6.1.2. Query Categorization

The best fitting available dataset coming from the query categorization background is perhaps the one used in the ACM KDD CUP 2005 for Query Categorization. This dataset contains 800 queries with labels (categories of the Open Directory Project) from three labelers in addition to 111 sample queries with labels. The number of unlabeled queries is 800.000. The number of target categories is 67.

This dataset exhibits different linguistic properties than free-text image annotations. High portion of the queries can be viewed as noise, there are queries such as “a”, “all”, number of queries are also not nouns and most queries have single or two words. No context is provided globally as the queries are unrelated. Finally, target labels are not provided in terms of a single term, but rather as a category of the Open Directory Project. The advantage is the focused nature of the queries – each query can be viewed as a maximum one entity, therefore there is a direct link between the entity and the label.

6.1.3. Image Classification

There is a number of research papers that try to use textual information attached to images in the image classification process. The datasets used typically contain single tags, rather than free text annotations. Perhaps the most commonly used one (e.g. [BJ07, GAS99]) is the Corel Stock Photo dataset.

Corel Stock Photo Dataset

The dataset consists of 5,000 images. Each 100 images are on the same topic, such as “Sunrises” and “Sunsets” or “Wild Animals”. Every image has a brief description of the scene (caption) and a list of objects that appear in the image (labels). An example of an image caption is “Man And Boy Fishing Mountain” while “Tree, People, Mountain, Water” are the labels. Overall 371 words are used as labels in the collection [BJ07]. Both the size and nature of this dataset would be fitting the entity classification task. The fact that images are partitioned into contextually similar groups is also an advantage.

The problem with this dataset is that there is no direct link between individual entities in the caption and the labels. In the example given above, the missing information are entity-label pairs {Man, People}, {Boy, People}, {Fishing, Water} and {Mountain, Mountain}. The dataset is also not freely available.

Yahoo News Dataset

Paper [DM07] uses 1700 randomly drawn images and image captions from Yahoo News. Not all of these images were used for evaluation, the authors mention manually selecting and annotating 100 image annotations with 50 used for training and 50 for testing. The dataset contains in average 15 entities per annotation. The annotations typically consist of several sentences.

While the nature of the dataset is close to the ideal dataset for the research presented in this dissertation, there are two problems why it was not used. First, randomly drawing images does not produce a dataset where a cross-image disambiguation could be used, since images cannot be expected to share a common context domain (not considering the rather generic "news" context). Second, probably for copyright issues the specific set of images and image captions used by [DM07] was not made available.

Israeli Images Dataset

This dataset was introduced in [BJ07]. It consists of 1823 images downloaded from `http://www.IsraelImages.com`. The context of these images is Israel scenery and society. The images are grouped into 11 categories: Birds, Desert, Flowers, Trees, Food, Housing, Christianity, Islam, Judaism, Personalities and Symbols. Each image has a 1 to 18 words long annotation. It should be noted that many image annotations are repeating.

According to paper [BJ07], this dataset is available to the research community, but the images can no longer be retrieved. The list of the images along with image captions and categories is still available.[1] Although the labels (categories) are not provided on the per entity level, but only on the per image level, in most cases the annotation contains only one entity. As a consequence, annotations can be thought of as entities and the provided image labels as classes. Example image annotations: "Young Pelican" (Bird), "Pigeons Standing On Electric Wire" (Bird), "Ramat Hanadiv Gardens In Zichron Yaakov" (Trees). The main issue with this dataset is that the image category was apparently assigned based on the image content, rather than based on the image annotations. For example, annotation "Wadi Besor" is annotated as Tree.

Czech Traveler Dataset

Inspired by the Israeli images dataset, we created a dataset based on a collection of 1,276 images taken by a professional photographer during trips to Albania, Corsica, Romania, Slovenia and Ukraine. These images have short textual annotations consisting of 1 to 10 words saved in images' EXIF data. Out of the annotations we extracted 103 unique annotations. The image annotations were broken into entities and these entities were assigned a label (class). These entities were not pruned for uniqueness. For example, the 'landscape' entity appeared 7 times. An overview of the size of the dataset is given by Table 6.1.

While this dataset originated already in 2008 [KCN+08a] within this dissertation we revised the dataset. This resulted in a removal of several entities: e.g. we removed entity "Church of Our Lady of Mercy in Buje" because this is the only entity which was incorrectly split into multiple entities (Church, lady, mercy, Buje) using noun chunker available in the GATE

[1] `http://www.cs.umass.edu/~ronb/datasets/info.txt` [Retrieved on 11 June 2012]

number of images	1,276
unique annotations	103
entities	186
labeled entities	184
labeled entities (to 9 classes) with inter-annotator agreement	**143**
unique entities	151
named entities	101
unique named entities	76
unique entities with inter-annotator agreement	113
entities for which not even the head was mapped to WordNet	**47**
unique entities for which not even the head was mapped to WordNet	41
entities for which not even the head was mapped to WordNet among the 143 entities with inter-annotator agreement	30

Table 6.1.: Entity statistics – Czech Traveler dataset, subsets used in the experiments are listed in bold.

framework. Further, we removed several entities which were not salient (entity "front") extracted from annotation "Ukainian girls chatting in front of fountain". Several typos were corrected, for example "Neoclassical Pallace in Buje" to "Neoclassical Palace in Buje". We also completely removed entities annotated as "event" because there were only two such entities.

As a result, we obtained 186 entities, 101 (54%) out of which are named entities (proper nouns).

The following set of nine WordNet nouns is used as the set of nine classes: $C = \{$*natural object, artifact, vehicle, geological formation, structure, organism, water, vegetation, landscape*$\}$. This selection was motivated by the needs of the image classification task in [KCN+08a] (classes *sand* and *event* were removed because they were not represented in ground-truth).

The dataset was annotated by two annotators in 2008. For each entity (noun phrase), the semantically closest concept was assigned. The annotators were allowed to use Wikipedia. Even then, for two entities, *Jetee du Dragon* and *Syri i Kalter*, the annotators were both unable to assign a label. The inter-annotator agreement was 77%. The resulting dataset containing 143 entities is used for evaluation of the SCM and BOA algorithms. The THD algorithm is evaluated on 47 entities for which not even the head noun could be mapped to WordNet. It should be noted that not all of these 47 entities are included among the 143 entities with inter-annotator agreement.

The annotations and the list of all 186 entities are given in the Appendix in Table D.1-Table D.3. The 143 entities with inter-annotator agreement are listed in Table D.4 - Table D.6. The 41 entities used for THD experiments were listed in Table 5.3.

Note that a reference to the "Czech Traveler dataset" in this dissertation always refers to the subset of 143 entities with inter-annotator agreement for SCM and BOA evaluation, and to the subset of 47 unresolved entities to WordNet for THD evaluation.

6.1.4. Hypernym Discovery from Wikipedia

To date, there is a very limited number of work on hypernym discovery from Wikipedia, which is also reflected in the availability of datasets. Even the recent paper [LLM11] introduces its own gold standard dataset. This dataset contained 4,000 tokens extracted from German Wikipedia (mostly first article sentences) in which 450 hypernyms were hand-tagged. This dataset could not be used because it was not made publicly available in addition to being German.

6.1.5. Word Similarity Computation

The most widely used benchmark dataset in the WSC area is the freely available[2] WordSim353 collection [FGM+02].

WordSim353 dataset

It contains two sets of English word pairs containing 153 and 200 word pairs along with similarity judgments assigned by 13 and 16 human subjects respectively. The judgment ranges from 0 (totally unrelated words) to 10 (very much related or identical words). The disadvantage of this dataset from the point of view of the entity classification task is that although practically all words can be interpreted as entities (nouns), they lack global context and named entities are virtually not represented in the collection.

This dataset can be further partitioned into two gold standard datasets: similarity and relatedness dataset [AAH+09]. The **similarity dataset** contains pairs of words considered as similar (synonyms, antonyms, identical, hyponym-hyperonym) and unrelated pairs (pairs with no clear relationship and with similarity equal or below a certain threshold). The **relatedness** dataset contains pairs considered as meronym-holonym, with no clear relationship and with a human average similarity greater than certain threshold and the unrelated pairs. The number of pairs in the similarity dataset is 203 and the number of pairs in the relatedness dataset is 252.

Out of the 437 unique words in the dataset, only 7 words cannot be directly mapped to a WordNet noun synset. Since mapping all words to WordNet is desirable for both SCM and BOA algorithms, for these 7 words we created a mapping manually by selecting a replacement word.

These mappings are with one exception straightforward: replacing the word with its singular form (media, children, earning), using a gerund instead of a verb (eat → eating, live → living) or vice versa (defeating → defeat). The one exception is the named entity Maradona, for which we use the THD result and map it to "footballer". We call the dataset where these replacements summarized in Table 6.2 have been made the WordSim353-WNaligned dataset.

The original WordSim353 dataset is available at `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/` and the split between the relatedness and similarity datasets at `http://alfonseca.org/eng/research/wordsim353.html` ([Retrieved June 11, 2012]). The original WordSim353 dataset is also reprinted in Appendix C.

[2] `http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/` [Retrieved on 11 June 2012]

Table 6.2.: Words that had to be replaced in WordSim353-WNaligned as compared to WordSim353 to ensure that all words are mappable to WordNet

WordSim353 original	WordSim353 aligned
media	medium
children	child
live	living
Maradona	footballer
eat	eating
earning	earnings
defeating	defeat

6.1.6. Word Sense Disambiguation

The WSD problem can be viewed as a more general task than the NER problem as explained in the introduction. A good overview of knowledge sources and evaluation datasets for WSD is given in [Agi07]. Historically, one of the most common systems used is WordNet. WordNet was in detail covered in Chapter 4.

Semcor is a semantically tagged subset of the English Brown Corpus [FK83] containing 360,000 words out of the roughly 1,000,000 words in the Brown Corpus. Semcor was created by the WordNet research team [MLTB93]. The disadvantage of this approach is the fact that this corpus is only about one third of the size of the Brown corpus and covers only about 25% of the WordNet noun senses.

Senseval (`http://www.senseval.org`) is a series of evaluation exercises for WSD. Five Senseval contests were held to date. The first Senseval contest focused on a limited number of generic words across different parts of speech. For nouns, only 15 generic nouns for the English *lexical sample task* such as "accident" or "float" are present [KR00]. For Senseval 2, the general character of the training senses for the lexical sample task is similar to Senseval 1. Senseval 2 and 3 also feature the *all-words task*, where the aim is to disambiguate all words, rather than a sample of selected words. In Senseval 3 approximately 5,000 words of coherent Penn Treebank text are tagged with WordNet 1.7.1 tags. Unfortunately, the selected text contains virtually no named entities. The generic character of words covered applies to all Senseval WSD tasks, including the following Senseval 2007 and SemEval 2010 "Word Sense Disambiguation on a Specific Domain" task. A generic set of words is clearly not suitable for our entity classification problem.

6.2. SCM on WordSim353 dataset

The fact that almost all entities in the WordSim353 dataset are directly mappable to WordNet concepts makes it a suitable choice for evaluating the performance of the SCM without influence of the THD phase. This in fact translates into computation of similarity of words using WordNet similarity measures. An illustrative example result for this task is depicted at Table 6.3.

The Spearman correlation coefficient is computed for the original WordSim353 dataset and

for WordSim353-WNaligned dataset. All the 7 word pairs that feature any of the entities listed in Table 6.2 have similarity 0 for the purpose of computing the Spearman correlation coefficient for WordSim353. For each measure we give two values depending on the way synsets representing the word are selected (refer to Subs. 4.3.1 and Subs. 4.3.3).

Table 6.3.: An illustrative WSC computation result on a fragment of the WordSim353 dataset. The word pairs are sorted according to the computed value, the ground truth similarity /average of similarity ratings assigned by human evaluators/ is also shown.

word 1	**word 2**	**truth**	**result**	**word 1**	**word 2**	**truth**	**result**
cash	money	9.15	12.693227	professor	student	6.81	6.444655
stock	company	7.08	8.980284	Market	stock	8.08	6.03275
Money	bank	8.12	8.6386455	life	stock	0.92	2.1296652
forest	wood	7.73	8.4673136	jaguar	stock	0.92	2.0128817
keyboard	computer	7.62	8.0023444	cucumber	professor	0.31	2.0075121
Butter	bread	6.19	6.837023	egg	stock	1.81	1.902016
radio	television	6.77	6.754272	radio	media	7.42	1.7957932
Potato	cucumber	5.92	6.7131314	stupid	smart	5.81	1.7427614
paper	book	7.46	6.6186313	live	stock	3.73	0.8057597

The results for all the individual WordNet similarity measures implemented in JWSL are given by Table 6.4, the results for JWNL by Table 6.5 and averages over all the measures across both libraries are given by Table 6.6.

Table 6.4.: WordNet similarity measures implemented in JWSL on WordSim353 dataset and WordSim353-WNaligned

Sense selection	Synset Similarity Maximization (SSM)				Most Frequent Sense (MFS)			
Measure	Resnik	JCn	Lin	P&S	Resnik	JCn	Lin	P&S
Original	0.33	0.31	0.33	0.33	0.32	0.32	0.32	0.32
Aligned	0.34	0.32	0.34	0.33	0.32	0.32	0.32	0.33

As expected, it can be seen that the results for WordSim353-WNaligned are consistently slightly higher than for WordSim353. An encouraging observation from Table 6.6 is that the geometric average for all measures yields better result, albeit only marginally, than any of the individual measures. Another interesting point is that the performance of the individual similarity measures is quite consistent across libraries with the exception of the Jiang and Conrath (JCn) measure significantly underperforming in JWordnetSim with the Most Frequent Sense option. The best overall results are obtained with a mix of strategies.

In our experimental setup, the JWordnetSim IC values come from file ic-bnc-resnik-add1 (British National Corpus, Resnik Counting, smoothing) – refer to Subs. 4.3.1.

Table 6.5.: WordNet similarity measures implemented in JWordnetSim on WordSim353 dataset and WordSim353-WNaligned

Sense selection	SSM		MFS	
Measure	Lin	JCn	Lin	JCn
Original	0.32	0.30	0.33	0.23
Aligned	0.33	0.31	0.33	0.24

Table 6.6.: WordNet similarity measures implemented in JWordnetSim and JWSL aggregated by geometric and arithmetic average on WordSim353 dataset and WordSim353-WNaligned

	geometric				arithmetic			
JWSL synset selection	SSM	MFS	SSM	MFS	SSM	MFS	SSM	MFS
JWordnetSim synset selection	MFS	MFS	SSM	SSM	MFS	MFS	SSM	SSM
Original	0.35	0.32	0.32	0.34	0.34	0.26	0.32	0.34
Aligned	0.35	0.33	0.33	0.35	0.35	0.33	0.33	0.34

noun phrase	hypernym	extr type	correct	annot 1	annot 2	SCM
Durau	resort	THD	1	structure	structure	landscape
Dolphin	dolphin	Direct	–	organism	organism	organism
ruins	ruin	Direct	1	structure	structure	structure
Glagolitic Alley	Alley	Head	1	vegetation	artefact	artefact
Albanian Guide Kamil	statesman	THD/Head	1	organism	organism	organism

Table 6.7.: Sample results of SCM/THD on the Czech Traveler dataset and comparison with annotators.

6.3. SCM on Czech Traveler Dataset

For practical reasons, the experiment description is divided into two parts, which corresponds to steps of the SCM algorithm. Subs. 6.3.1 is devoted to evaluation of the task of mapping noun phrases to WordNet synsets. Subs. 6.3.2 gives the results for WordNet similarity computation in SCM. An illustrative example for this task is depicted at Table 6.7.

6.3.1. WordNet Mapping

The goal of this experiment is to evaluate the performance of using SCM to map entities to WordNet synsets. The Czech Traveler dataset and live English Wikipedia as of September 8, 2011 was used. For each entity the author assessed whether the sense an entity is mapped to is its correct WordNet counterpart, synonym or its close hypernym. In the dataset, there are 47 entities (41 unique)[3], which cannot be mapped to WordNet. THD returns a WordNet mapping for 41 entities. The following entities were not mapped to WordNet: (Skanderbeg, Place St Nicolas, Eski Kermen, Istarske Toplice, Chufut-Kale, Khersones).

A detailed analysis of THD results is given in Subs. 5.4. THD produced a correct WordNet mapping for 29 entities (27 unique).

Overall, THD retrieved a correct hypernym for $29/47 = 62\%$ of entities. Note that these are hard cases – for these entities, SCM failed to map even the the head noun to WordNet.

6.3.2. WordNet Similarity Computation

This experiment uses entities mapped to WordNet using lemmatization, replacement of spaces and THD. The complete list of WordNet mappings is given in the Appendix in Table D.4 - Table D.6.

Entities are classified into the following set of classes: $C = \{$*natural object, artifact, vehicle, geological formation, structure, organism, water, vegetation, landscape*$\}$. For this classification, the ground truth created by two annotators is available.

The results for all the individual WordNet similarity measures implemented in JWSL are given by Table 6.8, the results for JWNL by Table 6.9 and averages over all the measures across both libraries are given by Table 6.10.

Using only the 143 entities with inter-annotator agreement on the label, SCM correctly classified 74% of entities using the best performing configuration, which was all measures in both libraries using synset similarity maximization. Due to missing WordNet mapping, 6

[3]The duplicate entries are Gjirokaster 2x, Jezersko, Khersones 2x, Qeparo

entities were not classified. For comparison, a baseline classifier, which would assign the most frequently occurring class "structure" to all entities, correctly classified 32% of entities.

Table 6.8.: WordNet similarity measures implemented in JWSL on Czech Traveler dataset

Sense selection	Synset Similarity Maximization (SSM)				Most Frequent Sense (MFS)			
Measure	Resnik	JCn	Lin	P&S	Resnik	JCn	Lin	P&S
	0.66	0.56	0.73	0.55	0.60	0.38	0.55	0.38

Table 6.9.: WordNet similarity measures implemented in JWordnetSim on Czech Traveler dataset

Sense selection	SSM		MFS	
Measure	Lin	JCn	Lin	JCn
	0.66	0.55	0.62	0.45

Table 6.10.: WordNet similarity measures implemented in JWordnetSim and JWSL aggregated by geometric and arithmetic average on Czech Traveler dataset

	geometric				arithmetic			
JWSL synset selection	SSM	MFS	SSM	MFS	SSM	MFS	SSM	MFS
JWordnetSim synset selection	MFS	MFS	SSM	SSM	MFS	MFS	SSM	SSM
	0.67	0.55	0.74	0.59	0.71	0.56	0.74	0.59

6.4. BOA on WordSim353 Dataset

This experiment will evaluate the effect of selected parameters on the classification performance as measured by Spearman correlation coefficient with WordSim353 ground-truth.

Table 6.12 gives an overview of the setup. The first two columns give names and default values for all parameters involved in the experiments. Each of the remaining columns describes one experiment batch. An experiment batch consists of several series of experiments (BOA classifier runs). The column describing an experiment batch lists the values of parameters that differ from the default values. If the value of the parameter applies only to one series in the batch, it is marked as so. Finally, one parameter within the column is marked with an asterisk. This parameter is varied – the different values of this parameter are plotted on the x-axis in the resulting graphs. The following notation is used to describe the experimental setup in Table 6.12:

- *empty cell* – the default value from column 2 is used,

- * – value of this parameter is varied in the experiment
- other value y – y is a value for the parameter different from the default for all series within a graph (experiment batch). The following abbreviations are used:
  - NA – Not Applicable
  - n – no, y – yes
  - t – true, f – false
  - m – mostsim, f – firstn
  - C – Custom Aggregator 1, P – Custom Aggregator 2, W – Weighted Geometric Average
  - c – cosine similarity, d – dot product
  - z – zero (crawling depth = 0)
  - k – thousand
- xy – where x is the number of the series within the graph and y is a value for the parameter different for this series than is the default. The remaining series have the default value.

The results of the experiments are plotted in Fig. 6.1-Fig. 6.18. Above each graph, there is a short description of the experimental setup in terms of differences from the default values listed in the second column of Table 6.12. The values of the parameters that are stable across all experiments, but have impact on the results are present in Table 6.11.

EntityClassifierTrainingConfig – Global parameters	
stopWordListPath	stopwordlist.txt, this file contains compilation of publicly available stop-word lists – 838 entries
EntityClassifierSearchConfig	
pathToFileWithLuceneArticleKeys	searchkeys.csv, this file contains automatically generated mapping of WordSim353 entries to Wikipedia as listed by Table C.6 and Table C.7
JWNL Config	
infoContentFileName	ic-bnc-resnik-add1.dat
jwnlinitPath	map_propertiesWN20.xml
Common Wordnet Config	
Synset selection	Synset Similarity Maximization
roundToZeroIfUnderWordnetSimThreshold_level$x$ (T_l^{low})	0.0
roundToOneIfAboveWordnetSimThreshold_level$x$ (T_l^{high})	1.0
Wordnet Measure as Wordnet aggregate component	
WeightingFactor	1.0
Common Modality Config	
WeightingFactor (W_m)	1.0
WeightFactor_level$x$ ($W_{m,l}$)	1.0
Common Term Weight Vector Config	
WeightFactor_level$x$ ($W_{m,l,t}$)	1.0

Table 6.11.: Constant parameters in BOA experiments (subject to normalization)

Table 6.12.: Overview of BOA experiments

parameter	default	e1	e2	e3	e4	e5	e6	e7	e8	e9	e10	e11	e12	e13	e14	e15	e16	e17	e18
similarityFunction	dotProduct															c		1c,2c	
same CATegory modality	y	NA	3n,4n		2n,3n	2n,3n	NA	NA			2n,3n		2n,3n					4n	
– aggregator	customAgg 1																1C/2P/3W		
IN-link modality	y	NA	2n,4n		1n,3n	1n,3n	NA	NA			1n,3n		1n,3n						
– aggregator	customAgg 1																1C/2P/3W		
OUT-link modality	y	NA	2n,3n		1n,2n	1n,2n	NA	NA			1n,2n		1n,2n					4n	
– aggregator	customAgg 1																1C/2P/3W	1W	
maxTermVectorLength	200000	*	*	*		20k	*	*				*	20k	20k	*	*	*	*	*
discardTermsNotInWordnet {**t**rue, **f**alse}	t	1f,2t	f		f		f		f			1t/2f/3t					1f	1f 3f	
crawlingDepth	1	z						z										1z,2z	1z,2z,3z
maxLinksToFollow	20				*	*			*	*	*			*					
articleSelectionStrategy {**f**irstn, **m**ostsim}	f	NA								m	m	1m/2m/3f	m	m	m	m	m		
IDF-ALL used	no						2y	2y	2y	2y				1y 3y 4y	1y 3y 4y	1y 3y 4y	y	2y 3y 4y	y
IDF-BOA used	no						3y	3y	3y	3y				2y 3y 4y	2y 3y 4y	2y 3y 4y	y	3y 4y	y
Wordnet WeightingFactor	1																		
– roundToZero..._level0	0																		
– roundToOne..._level1	1																		
JWordnetSim JCn used	no			1y		4y												3y,4y	2y,3y,4y
JWordnetSim Lin used	no			2y															
JWSL JCn used	no			3y															
JWSL Lin used	no			4y															
JWSL Pirro Seco used	no			5y															
JWSL Resnik used	no			6y															
JWSLJWNL all	no			7y		5y								4y	4y	4y	y		

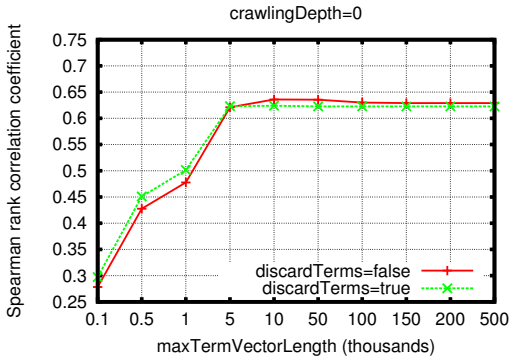

Figure 6.1.: Experiment 1 – only entity article: WordNet term pruning on and off

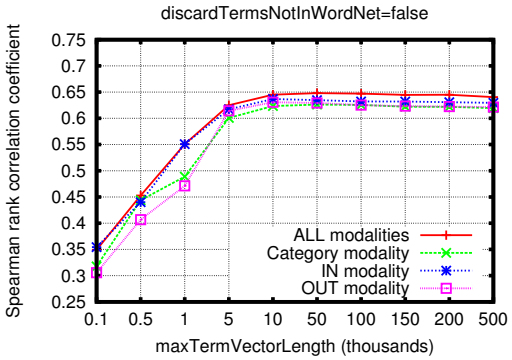

Figure 6.2.: Experiment 2 – modalities: WordNet term pruning off

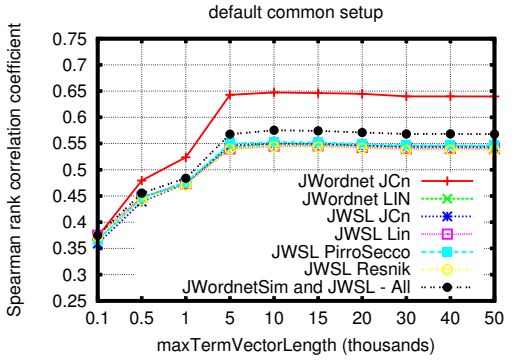

Figure 6.3.: Experiment 3 – WordNet similarity measures as term-weighting function

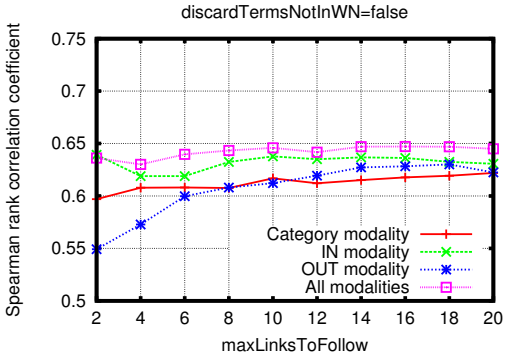

Figure 6.4.: Experiment 4 – modalities: impact of number of (randomly selected) links

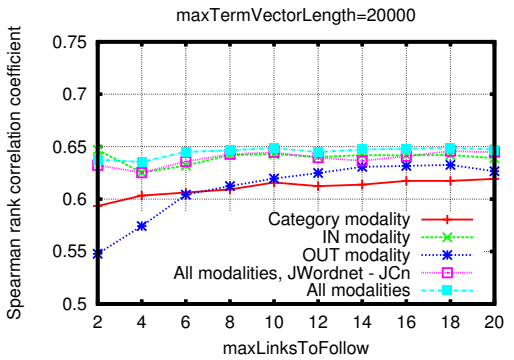

Figure 6.5.: Experiment 5 – modalities: impact of number of (randomly selected) links, limited vector length

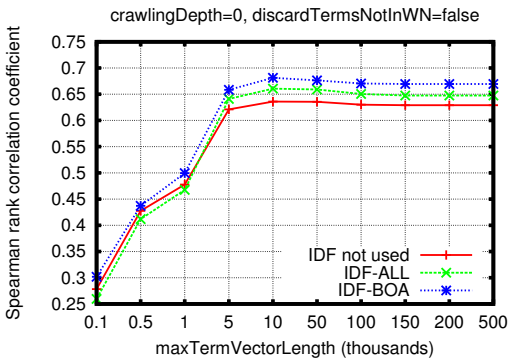

Figure 6.6.: Experiment 6 – IDF: only entity article, WordNet term pruning set to off

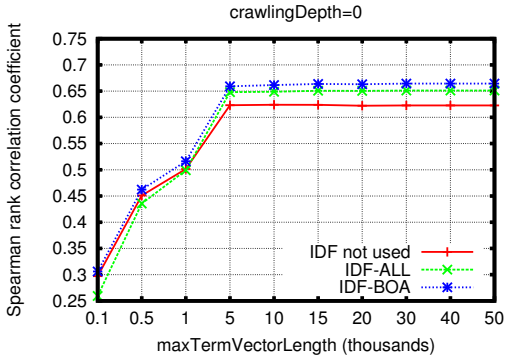

Figure 6.7.: Experiment 7 – IDF: only entity article

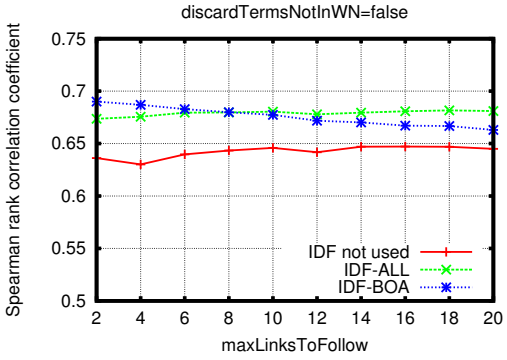

Figure 6.8.: Experiment 8 – IDF: level 1 under limited vector length

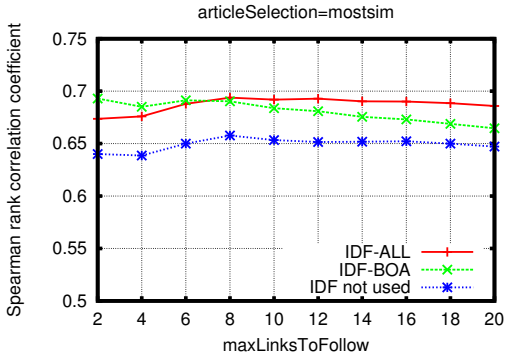

Figure 6.9.: Experiment 9 – IDF: most similar article selection

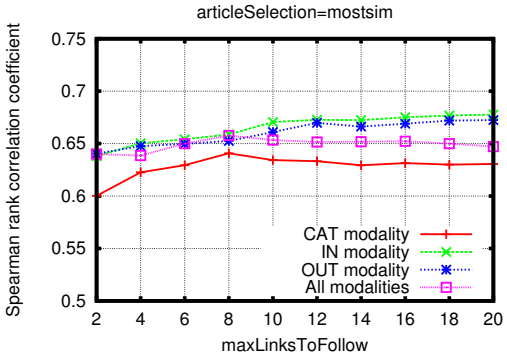

Figure 6.10.: Experiment 10 – modalities: most similar article selection

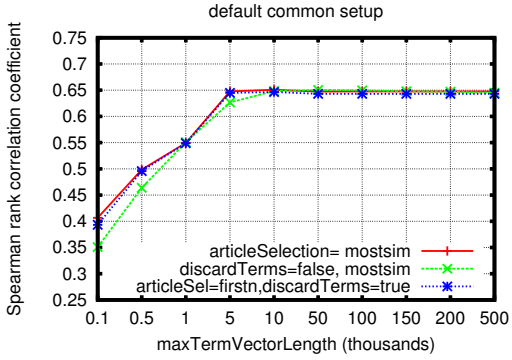

Figure 6.11.: Experiment 11 – combinations of article selection with WordNet term pruning on/off

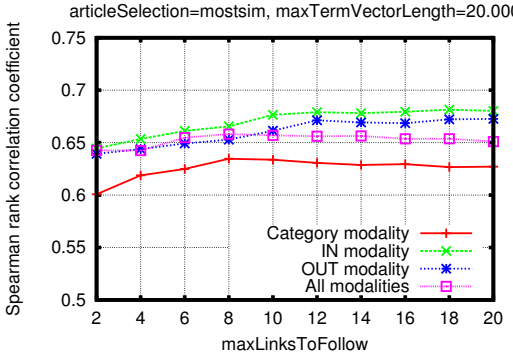

Figure 6.12.: Experiment 12 – modalities: most similar selection and limited vector length

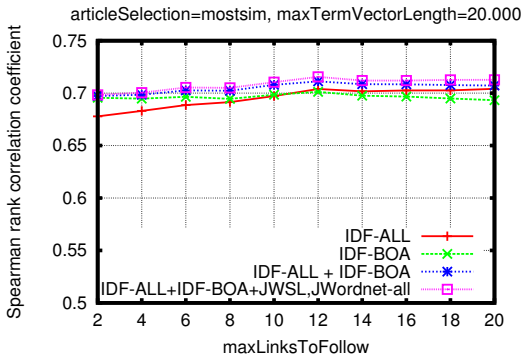

Figure 6.13.: Experiment 13 – IDF+All WordNet measures: mostsim article selection and limited vector length

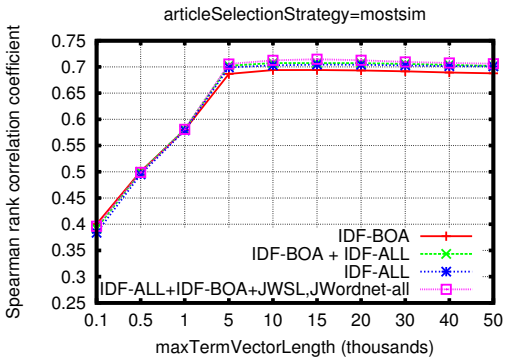

Figure 6.14.: Experiment 14 – IDF+All WordNet measures: most similar article selection

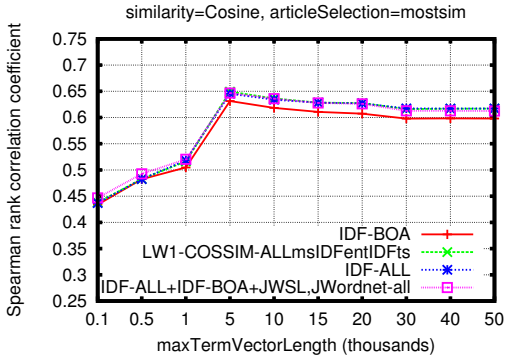

Figure 6.15.: Experiment 15 – similarity function / cosine vs dot product

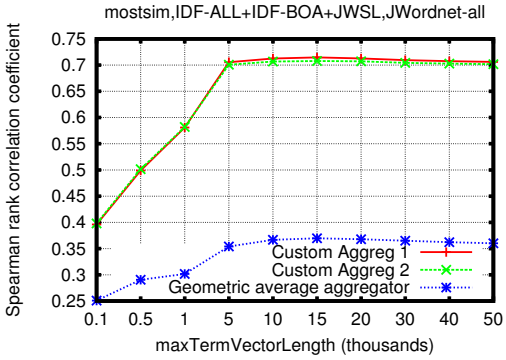

Figure 6.16.: Experiment 16 – aggregator comparison

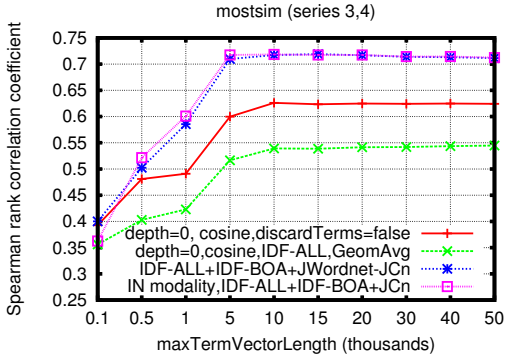

Figure 6.17.: Experiment 17 – baseline and best BOA configurations

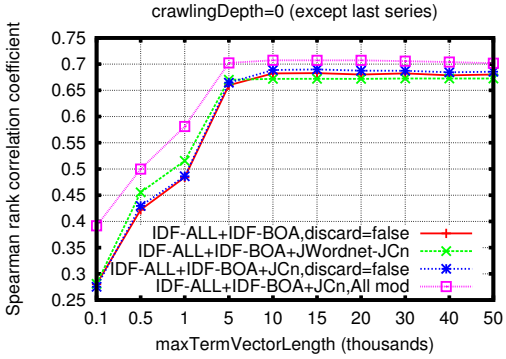

Figure 6.18.: Experiment 18 – best BOA configurations for level 0 and level 1

The Experiments 1-18 presented in Fig. 6.1 – Fig. 6.18 can be used to derive the impact of individual features on the performance of the BOA algorithm:

Crawling setup

The *maxLinksToFollow* parameter sets how many links related to article on level l will be used to create lm band for level $l+1$. If the number of available links for an article is larger than the *maxLinksToFollow* threshold, articles are selected according to the *articleSelectionStrategy*. With *firstn* strategy, first *maxLinksToFollow* links as encountered is processed (this is close to a random order), with the *mostsim* strategy, articles are selected according to their textual similarity.

Experiments 12 and 5 demonstrate that the *mostsim* article selection is superior to the *firstn* article selection. The associated increase of the correlation coefficient is about 0.03.

If *firstn* article selection is in place, Experiment 4 shows that the maximum correlation of 0.65 is attained with 15 links, using more links does not deteriorate performance. Experiment 5 shows that if WordNet-based term pruning is turned on and the term vector length is limited to 20,000 terms, the peak correlation of 0.65 remains the same, but it is obtained already with 10 links.

Modality performance

The peak performance obtained with the simplest setup – only entity article (*crawlingDepth*=0) and term frequency as the sole term-weighting function – is 0.64 as demonstrated in Experiment 1. Interestingly, Experiment 2 shows that adding random (*firstn*) 20 articles from any other modality does not improve this result. Only using all modalities simultaneously produces a result of 0.65. However, all these experiments were using *firstn* article selection. In Experiments 10 and Experiment 12, which use *mostsim* article selection, the best results are obtained with only the in-link modality.

Term Vector Length

The *maxTermVectorLength* parameter sets the maximum number of terms that will be used to create the BOA term vector. If the number of words encountered in all the articles involved in training is larger than this threshold, the words are sorted according to their BOA frequency and *maxTermVectorLength* words with highest BOA frequency is retained. BOA frequency is the number of entities for which the term occurs at least once.

Concerning term vector length, almost all experiments that varied the *maxTermVectorLength* parameter (Experiments 1-3,6,7,11,14-18) indicate that best results are obtained with term vector length of around 10,000. Using shorter vectors has severe negative impact on the correlation coefficient, while using longer term-weight vectors does not have any effect. We suggest the following conjecture to explain this: in shorter term-weight vectors important discriminative words are lost.

In contrast, words above this threshold have low BOA frequencies and they can thus be considered as "white noise". Experiment 15 is the only exception, the peak performance is obtained already at 5,000 terms, and what is perhaps more important, it drops afterwards. In comparison with Experiment 14, Experiment 15 differs only in the similarity function – cosine similarity instead of dot product, which is used in all other experiments. The reasons are discussed in the next paragraph.

Term Vector Comparison

Our implementation features two vector comparison (similarity) functions – cosine similarity and dot product. Their performance can be compared by contrasting the first series from Experiment 17 with first series in Experiment 1. In this simple setting, when only the term frequency term-weight vector is involved, the best performance of cosine similarity lags behind the best performance for dot product by about 0.01. In presence of multiple term-weight vectors as evaluated by Experiments 14 and 15, the dot product gives even better performance with increase in correlation coefficient exceeding 0.05. Furthermore, these experiments show that the performance of cosine similarity deteriorates with increasing dimension, which does not happen for the dot product.

The dot product, as the best performing measure, was therefore used as a default for other experiments.

Term Vector Aggregation

The implementation offers three possible aggregators for combining the individual term-weight vectors such as TF or IDF-BOA into one term-weight vector per modality. The aggregation is performed on a per component basis. The challenge is that WordNet-based term-weight vectors are sparse – they may have zero values on some positions, however, it is not desirable for the result of the aggregation to be zero, when other term weights have non-zero values on the corresponding positions. Such behaviour is manifested e.g. by geometric average.

Experiment 16 shows that this is indeed an issue. If geometric average is used as an aggregator, the performance is mediocre. The two aggregators that we have proposed provide markedly better performance providing an increase of 0.3. It should be noted that we assume that the impact would likely be much smaller should all aggregated vectors be dense, without non-zero entries. This corresponds to the WordNet term-weight vector not being used.

The best performing Custom Aggregator 1 is used as default in other experiments.

Inverse Document Frequency

Experiments 6 and 7 demonstrate that using IDF term-weight vector improves performance. IDF-BOA is slightly superior to IDF-ALL, however both give results over TF-only baseline. These experiments are limited to crawling depth 0. Experiment 8 carried out under crawling depth 1 shows IDF-BOA giving initially better results than IDF-ALL. However, performance of IDF-BOA quickly deteriorates with increasing *maxLinksToFollow*, while the performance of IDF-ALL remains virtually stable. Nevertheless, all three experiments carried out under the *firstn* article selection indicate that IDF improves performance over TF-only baseline and that the contribution of IDF-ALL can differ from contribution of IDF-BOA by as much as 0.02.

An interesting perspective gives a comparison of Experiment 11 with Experiment 14. These two experiments allow to compare IDF and non-IDF setup under the *mostsim* article selection. The results indicate that using IDF improves the correlation coefficient by about 0.05. In the *mostsim* context, the difference between IDF-BOA and IDF-ALL is diminished to around 0.01. Nevertheless, using both IDF-based term-weight vectors simultaneously does provide a small performance increase, as best shown by Experiment 13.

WordNet

Experiment 1 shows that using **WordNet as a positive term list and lemmatizer** (through *discardTermsNotInWordnet* parameter set to true) has slight negative impact on peak performance if crawling depth is 0. This is compliant with the intuition that an article directly describing the word will not contain much noise words. Experiment 11 shows that for crawling depth 1, setting *discardTermsNotInWordnet* to true has virtually no impact. The same experiment also shows that this result holds for both *firstn* and *mostsim* article selection. However, using WordNet as positive term list consistently improves results for shorter term vector lengths. This can be used to shorten execution time at the expense of a drop in the classification quality.

WordNet similarity measures can be used to create **WordNet term-weight vector**. The best performing WordNet similarity measure for this purpose is Jiang and Conrath (JCn) from the JWordnetSim library as shown in Experiment 3. The increase in performance outperforms even a weighted average of all measures. Nevertheless, Experiment 5 shows that even the best performing measure does not improve the baseline created by all modalities. Again, these results apply for *firstn* article selection.

In contrast, Experiments 13 and 15 show that using WordNet term-weight vector (again average of all WordNet measures) improves the performance over the baseline. These latter two experiments differ in a) *mostsim* used for article selection instead of *firstn* and b) at least one IDF-based term-weight vector used in addition to TF.

We also investigated the impact of the choice of the `infocontent` file on the results. The evaluation spanned 20 `infocontent` files provided by Ted Pedersen,[4] only `infocontent` files for compounds were excluded. This list includes files created from the Brown corpus, British National Corpus, Penn TreeBank, Semcor, and Complete Works of Shakespeare, each with several variations in IC computation. These files were individually tried for computing the WordNet term-weight vector in our best performing setup (in-link modality, IDF-ALL, IDF-BOA, JWordnetSim-JCn). The results indicate negligible (about 0.001) impact on BOA performance.

Best Performing Configuration

The standard use of bag-of-words approach involves term frequency, possibly TF-IDF, and cosine similarity as the vector comparison measure. Performance of the baseline setup was investigated in Experiment 17. It should be noted that our BOA implementation does not support multiplication used in TF-IDF as an aggregator, therefore in the experiment we used geometric average as the closest available aggregator. The best result obtained with this setup is 0.625 (0.55 without IDF). These baseline classifiers are compared with a full-featured BOA classifier involving all modalities in the distance of one link, which achieves performance of 0.7190. It should be emphasized that all experiments involve stop-word removal.

This best-performing setup is computationally expensive, especially due to high costs associated with crawling the Wikipedia in all three modalities. Using only the in-link modality, the performance is with 0.7185 virtually the same, however with smaller computational demands. As shown by Experiment 18, correlation coefficient of 0.69 can be obtained without any crawling, only by exploiting the text contained in the entity article. Using modalities ($crawlingDepth > 0$) thus improves the performance by around 0.03.

[4] `http://www.d.umn.edu/~tpederse/Data/WordNet-InfoContent-2.0.tar.gz` [Retrieved on 11 June 2012]

6.5. BOA on Czech Traveler Dataset

There are several principal options how BOA experiments on the Czech Traveler dataset can be carried out, which relate to different degrees of supervision the BOA algorithm allows:

- no labeled instances[5] provided for training phase with meta parameters set heuristically (e.g. based on results obtained on the WordSim353 dataset).
- no labeled instances with meta parameters determined by the genetic parameter estimation algorithm from a subset of the Czech Traveler dataset
- labeled instances are provided, with two-fold cross validation one half of the Czech Traveler dataset is used for training and the other half for testing. There are two variants of this "most supervised" run, depending on whether:
  - name of the target class is used as a training instance,
  - name of the target class is not used as a training instance.

These options translate into four BOA configuration setups. The cross validation setup is presented in Subs. 6.5.1 and the parameter estimation procedure in Subs. 6.5.2.

6.5.1. Repeated Two-fold Stratified Crossvalidation

The Czech Traveler dataset was partitioned into two stratified subsets, i.e. the proportion of all target labels was the same (with maximum absolute difference of 1) in both subsets. This random partitioning was repeated ten times, ten pairs of training and testing datasets emerged from this process. For experimental results listed in Table 6.13 involving cross validation (Experiments 3,4,7,8), the result is an average for these ten splits. For the remaining experiments 1,2,5,6, the results were obtained by running the experiment on the entire Czech Traveler dataset.

6.5.2. Parameter Estimation

BOA parameters were determined with the genetic parameter estimation algorithm using the training set from the first split produced for cross validation.

The base XML configuration file contained all available features in our BOA implementation: all modalities and supported crawling depth up to 1. For the training phase, all the implemented term-weight vectors were available in each modality: IDF-ALL, IDF-BOA and a WordNet aggregate term-weight vector over all implemented measures in JWSL and JWordnetSim libraries. For the test phase, only term frequency was available. The global parameters were also subject to optimization, most importantly the *discardTermsNotInWordnet* parameter, *maxTermVectorLength* and *similarityFunction*. Altogether there were about 100 features subject to optimization.

The genetic algorithm setup was as follows: maxGenerations = 50, populationSize = 60, maxGensWithoutImprovement = 5, mutationProb = 0.2. The genetic algorithm was run separately for each of the six experiments, where parameter estimation is involved.

[5] For each target class, there is technically always one "training" instance – the name of the class mapped to a Wikipedia article.

6.5.3. Results

The results are depicted in Experiments 1-4 in Table 6.13. The best result of 0.57 is obtained by a supervised run. Comparing this value with results for the SCM classifier in Table 6.10, it is clear that SCM provides with best value of 0.74 significantly better performance. Referring to Table 6.8 and Table 6.9, it is clear that the BOA result is surpassed by most single WordNet similarity measures. Additionally, SCM is computationally much less intensive: it does not require any training, and even classification is faster.

It is interesting to see BOA underperform SCM on the Czech Traveler dataset by quite a large margin, and the opposite result on WordSim353. We provide two plausible explanations:

- The nature of the *tasks* performed on the WordSim353 and Czech Traveler datasets is different. The setup of experiments on the WordSim353 dataset is quite special in the broader data mining context – there are no testing instances and target classes. Also, for a particular similarity result it is not possible to say whether it is good or bad. The overall performance is seen only from the rank correlation on the entire dataset. In contrast, the Czech Traveler dataset is conceived as a regular data mining dataset. There are eight target classes and 143 entities, each assigned one of the target classes.
- The nature of the entities involved is different. While WordSim353 contains almost exclusively general words, which can be mapped to information-rich Wikipedia pages, Czech Traveler dataset contains a large proportion of named entities with very brief Wikipedia pages.

In order to explore the significance of the second reason, a special WordNet mapped version of the Czech Traveler dataset was prepared. Entities in this dataset that could not be directly mapped to WordNet were replaced by their WordNet mappings. In other words, the entities were first mapped to WordNet and then to Wikipedia. The seed articles used are listed in the last column in Table D.4 - Table D.6. The results on this amended dataset are presented in Experiments 5-8 in Table 6.13. While there is some impact on performance, the improvement, if any, is not of the scale which was expected.

Table 6.13.: BOA Experiments on Czech Traveler dataset. Results are reported in terms of accuracy (correctly classified entities / all entities)

	Direct Wiki Mapping				**WordNet Mapping**			
Experiment number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
parameter estimation	no	yes	yes	yes	no	yes	yes	yes
training with cross validation	no	no	yes	yes	no	no	yes	yes
training with target classes	no	yes	no	yes	no	yes	no	yes
result	**0.29**	**0.55**	**0.57**	**0.58**	**0.31**	**0.55**	**0.44**	**0.42**

6.6. Final Assessment

In this section, we summarize the experimental results obtained with SCM and BOA algorithms across the Czech Traveler and WorsSim353 datasets. We separate the discussion

according to the algorithm used into two subsections.

6.6.1. SCM algorithm

Concerning the results of the WordNet similarity measures on the WordSim353 collection, the variations between individual measures were very small with all measures giving correlation coefficient between 0.31-0.34. There was one exception, in a particular setup consisting of Most Frequent Sense (MFS) assumption in combination with the JWordnetSim library, Jiang and Conrath gave only 0.23-0.24.

For the Resnik measure we have obtained virtually the same results as reported in [SP06]. The difference of around 0.01 might be partly attributed to the use of different Information Content values, since [SP06] do not report on what corpus they were computed from, and partly to the used method to measure correlation. Spearman correlation coefficient is employed in our research and in other more recent papers, while Pearson product-moment correlation is used by [SP06].[6] We are not aware of the results for Lin, Jiang and Conrath and Pirro and Seco on the WordSim353 collection that we provide being available in previous papers.

The SCM algorithm gave a comparatively better result on the Czech Traveler dataset. There, the difference in performance of individual measures was much larger. The best performing single measure was with 73% of correctly classified entities the Lin measure under Synset Similarity Maximization. The worst performing measure was with 0.38 accuracy again the Jiang and Conrath under the MFS assumption. The average for all measures yielded performance between 0.55 to 0.74.

6.6.2. BOA algorithm

An extensive evaluation of the BOA algorithm was performed. The experimental results reported in Sec. 6.4 summarize close to 1.000 runs of the BOA algorithm with different configurations. The results indicate which of the proposed features and parameters are most effective in producing similarity scores that are close to human:

- WordNet term-weight vector – the entries of this vector are values of Jiang and Conrath similarity computed with the JWordnetSim library between the term and the training entity.
- IDF-BOA – a term-weight vector based on a variation of the IDF measure. The same formula is used, but document frequency is replaced with "BOA frequency" – number of entities for which the term occurs at least once. IDF-BOA improves performance also if used alongside a term-weight vector based on the "regular" IDF measure.
- Custom Aggregator 1 – was proposed to aggregate multiple term weight functions, in context of the term weight functions in our BOA classifier, it provides superior performance to geometric average.
- Dot product on L1 normalized term-weight vectors – according to the experimental results, dot product on L1-normalized term-weight vectors consistently outperforms cosine measure. There is also a computational advantage to using dot product. The L1 normalization follows from our BOA model, so no extra operations are necessary. In contrast,

[6] The values of Pearson product-moment correlation are used interchangeably with Spearman rank correlation e.g. in [GM07].

cosine similarity implies L2 normalization, which has to be carried out as an additional operation.

- In-link modality – using articles that link to the article describing the word (entity article) provides best performance in comparison to using articles that are identified through links from the entity article, or articles in the same category. Perhaps surprisingly, the in-links even surpass an unweighted average of all the three implemented modalities. To achieve best performance, it suffices to select 10 to 20 linked articles that are most similar to the entity article.

- For best performance, it is sufficient to select 10,000 terms with the highest BOA frequency.

BOA classifier is a text-based method. The best performing text-based classifier on the WordSim353 dataset reported in [SP06] achieved only 0.20 Spearman rank correlation. This classifier used only the text of the articles, to which the words in the WordSim353 dataset were mapped to.[7] Our best-performing classifier that uses also only the text of the entity article (crawlingDepth parameter 0) produces correlation result of 0.69.

Comparison with results of state-of-the-art WSC algorithms that use other than text information from Wikipedia is also favourable. By involving also related articles, but no labeled data, BOA classifier achieves 0.72 Spearman rank correlation on the WordSim353 dataset. This result is thus placed in between Wikipedia Link Measure (0.69) and the best performing Explicit Semantic Analysis (0.75). Our best performing configuration left at their default values many parameters such as weights of levels, modalities and term-weighting functions. Our preliminary results indicate that proper adjustment of the W_m, $W_{l,m}$ and $W_{m,l,t}$ weights can further improve the results by more than 0.01.

Contrary to our expectations, BOA has mediocre performance on the Czech Traveler dataset. The best performing supervised setup correctly classified only 58% of entities, which compares badly with the 74% of correctly classified entities by the best SCM classifier. Further research is needed to identify the causes. It may be also the case that similar issues are encountered by other Wikipedia-based algorithms. As a first step for future work, we therefore suggest evaluation of ESA and WLM algorithms on the Czech Traveler dataset.

[7] A negligible improvement of 0.01 was observed when only the first sentence of the article was used.

7. Conclusions

This thesis aspired to develop a solution for the entity classification problem. In the introduction, we stated the following specific requirements:

- holistically address the entity classification problem:
  - extract entities from plain text,
  - accept user-defined set of target classes or no set of classes at all,
  - require no training set from the user,
- have results comparable with the state-of-the-art algorithms,
- disambiguate using global context,
- follow zeitgeist, but maintain (near) real time classification performance,
- use English as the target language, but allow for extensibility to other languages.

In the following, we will go through this list and show to what degree these points were met.

For the purpose of tackling the entity classification problem several algorithms were proposed and implemented. Sec. 7.1 shows that the designed solution completely addresses the entity extraction and classification problem. We can conclude that the performance of the proposed BOA algorithm is comparable with the state-of-the-art in the realm of Word Similarity Computation (WSC). Evaluation on the WordSim353 dataset, the gold standard in the WSC domain, shows only 0.03 difference from the state-of-the-art algorithm ESA. However, WordSim353 dataset is of rather academical nature. The BOA and SCM algorithms were therefore benchmarked also on a real-world Czech Traveler dataset. The results are summarized in Sec. 7.2.

An issue is the evaluation of the algorithms in the disambiguation task. We were not able to find a suitable dataset for this task, perhaps since classification of entities into a user-defined set of classes is not a fully established field. This is one of the reasons why the disambiguation algorithm proposed and developed for the use with our BOA remained experimentally unverified as discussed in Sec. 7.3. Sec. 7.4 highlights relatively low computational demands and the ability to work against live Wikipedia, which applies to both SCM and BOA algorithms. Sec. 7.5 gives account of the extensibility of the proposed algorithms from English to other languages. Sec. 7.6 lists minor contributions of potentially standalone applicability.

7.1. Addressing the Entity Classification Problem

This dissertation was motivated by a practical problem of using the text attached to images or videos to provide additional information to the image classifier. The algorithms and software developed within this dissertation were therefore designed and implemented with the intent to be used in a specific practical application, which required a complex approach and put constraints on software architecture as well as hardware requirements.

7.1.1. Extract Entities from Plain Text

In the course of the research it appeared that the entity classification is not an established discipline per se, although there are several closely bordering areas such as named entity recognition, word similarity computation and word sense disambiguation. In the thesis, we decided to focus solely on entity classification, putting entity extraction aside as a seemingly easier problem for which the ready-made approach based on the GATE ANNIE pipeline taken in the SCM implementation yields satisfactory results. Entity extraction problem is not addressed by our BOA implementation.

7.1.2. Unsupervised, Semi-Supervised Learning

If THD is used as a standalone classification algorithm, it performs *unsupervised classification*. The target classes are the hypernyms discovered.

The set of available target classes for SCM algorithm is determined by the set of words in WordNet. SCM does not have any training phase, nor does it require or can use any labeled instances. Instead, the algorithm uses knowledge previously inserted by experts into the WordNet thesaurus. Since SCM works with a user-defined set of target classes, it can be perceived as a *semi-supervised learning* algorithm.

The target class in the BOA algorithm is designated by a Wikipedia article. This article also constitutes the only required labeled instance, which is used as a seed to obtain other articles for the target class. The method optionally allows to specify multiple labeled seed articles for one class. BOA is a typical *semi-supervised learning* algorithm.

7.2. Performance

Algorithms devised within this dissertation did not yield improvement over the 0.75 correlation with human judgment achieved by ESA on the WordSim353 dataset, to the author's knowledge the best result by a single algorithm to date. However, using only the textual information from Wikipedia, our BOA algorithm run without any labeled data achieves correlation of 0.72. This not only exceeds the *text* baseline of 0.19-0.20 reported in [SP06], but also the Wikipedia Link Measure (WLM) [MW08], which has been with 0.69 correlation coefficient one of the best performing algorithms so far. In fact, BOA matches the performance of WLM by producing correlation of 0.69 by using only the text of the article defining the word, and not using any "bag" of articles at all.

The WordSim353 dataset, the standard benchmark in the WSC area, contains small number of named entities, with most entries in the dataset being generic nouns, it therefore provides a poor benchmark for the entity classification problem.

It is a matter of future work to thoroughly investigate the comparative performance of BOA to other algorithms on an entity classification rather than WSC dataset. The first results obtained on the Czech Traveler dataset indicate that the performance may not be as good as could be expected based on the encouraging results on the WordSim353 dataset. The best supervised run of the BOA algorithm correctly classified only 58 % of entities. In contrast, the results of the SCM algorithm were comparatively better with 74 % of correctly classified entities in the best performing setup. While the classifier in the SCM algorithm is based on existing WordNet similarity measures, we have contributed to this result through our THD algorithm, which allowed to map many of the input named entities to WordNet.

In a standalone evaluation of THD on Czech Traveler dataset the algorithm returned a correct hypernym for 62% of entities, for which not even the head noun could be mapped to WordNet in a straightforward way. It should be noted that a hypernym discovery algorithm, based also on our THD approach, was with good results evaluated on a German dataset within the recent paper [LLM11].

Out of the two algorithms proposed in this dissertation, BOA stands a comparison with the state of the art in the WSC context. Since most existing state-of-the-art WSC approaches use Wikipedia [GM07, MW08, SP06] (listed in the decreasing order of their performance on the WordSim353 dataset), this comparison is in our opinion fair, as BOA relies on the same knowledge source. It could be objected that since we did not rerun the experiments for other algorithms, different Wikipedia snapshots had to be inevitably used which might affect the results. Fortunately, the WordSim353 dataset contains general words, the Wikipedia entries for which, we assume, did not undergo evolution of the scale that could significantly affect the results.

7.3. Disambiguation

We noticed in our past work [CKN+08, KCN+08a] that object annotations tend to have common *global* context within the collection. Referring to the image use case, the collection of images classified could come from the same football match. To the best of the author's knowledge, existing WSC algorithms do not exploit the global context of the test set. A generic disambiguation algorithm was designed and implemented as part of BOA within this dissertation.

Unfortunately, due to the lack of an apt test dataset, the disambiguation algorithm was not evaluated. The WordSim353 dataset is not suitable for evaluating disambiguation, since (1) individual word pairs in the dataset do not share a common context, (2) applying the most frequent sense assumption results in incorrect disambiguation only for a fraction of entries. Although the Czech Traveler dataset was conceived also with the intent to evaluate disambiguation algorithms, it turned out that this dataset is also not suitable for this purpose, as the first sense is the best selection for almost all entities.

7.4. Following the Zeitgeist and Computational Requirements

It seems that as the performance of the existing methods increases, the demands on preprocessing increase, too. The first proposed Wikipedia-based WSC algorithm, WikiRelate!, is relatively efficient when it comes to preprocessing as it works only with the category hierarchy in addition to the text of Wikipedia articles that represent the two concepts compared. The best performing ESA method [GM07] involves parsing of all Wikipedia articles and extensive preprocessing [MW08]. The most favourable ratio between performance, and computational and preprocessing requirements is given probably by the WLM algorithm [MW08], which requires no preprocessing other than extraction of Wikipedia link structure and statistics on how often are anchors used to link to different concepts. This information can be obtained directly from Wikipedia XML dumps.

As around 1,000 new articles are added daily to Wikipedia, the reliance on intensive preprocessing may for topical entities turn as a disadvantage. The excessive preprocessing/computational costs of some algorithms was also noted in [MW08].

The SCM algorithm THD component runs in near real-time against live Wikipedia. It requires only one Wikipedia search and a subsequent download of the top matching article (several articles). A user of the algorithm therefore needs to wait only several seconds maximum (largely depending on the response time of the Wikipedia API) to get answer to a single query for a hypernym. The classification of a test entity with SCM takes generally less than one second provided that the test entity can be also directly mapped to WordNet and the number of target classes is not too large.

The BOA algorithm can in principle run against live Wikipedia as well, as it does not require any global knowledge. For BOA in minimum configuration, the requirements are the same as for THD: one Wikipedia search and the download of the top ranked article.

The articles in the out-link and same category modalities can be obtained from the entity article. The Meadiawiki "what links here" feature can be used to obtain the articles in the in-link modality. What concerns the term weight functions, all but IDF-ALL (the "classical" IDF) do not require information from articles outside the entity bag. Our experiments showed that the IDF-ALL measure can be with little impact on results replaced by IDF-BOA, which is computed only from articles in the bag. Also, IDF-ALL is a quite robust measure. Its values from an older Wikipedia snapshot will be likely available for virtually all words in any new Wikipedia article. Should a value not be available, it can be assumed that the document frequency for this word is 1 (the word appears in a new article for the first time) and the IDF-ALL value can be based on this estimated.

A simple training setup can take about one second to create a BOA representation for one target class. The time to create a BOA representation for a test entity is lower, because only term frequency needs to be computed. On the other hand, for test entity an extra time is taken by Wikipedia search, which maps it to a Wikipedia article. The time taken by classification, i.e. the similarity computation between the test entity and each target class, depends primarily on the selected BOA dimensionality, number and type of term-weight vectors, crawling depth, and the number of target classes.

To give a quantitative perspective, a complete run of the BOA setup on the WordSim353 dataset involving only the entity article using the IDF-ALL and IDF-BOA term-weight vectors takes about 20 seconds and produces correlation coefficient 0.68. An additional improvement of 0.01 to 0.69 can be obtained by plugging-in the JCn WordNet term-weight vector, with the execution time rising to 100 seconds (including a time intensive initialization of the memory-based JWNL store). The best performing configuration involving the in-link modality (crawling depth 1) with IDF-ALL, IDF-BOA and JCn WordNet term-weight vector takes about 20 minutes and produces correlation coefficient of 0.72.[1]

While BOA is not as low-cost as WLM, it still offers near real-time classification while providing a slight improvement of results on the academic WordSim353 dataset. Our conjecture is that the difference in the results will be greater in favour of BOA on real-world datasets, where entities are more likely to be associated with too few links for WLM to work reliably. We leave evaluation of this hypothesis for future work.

A potential technical advantage of our BOA implementation is that it operates on a standard Wikipedia Lucene index produced (with modification of two flags) by Mediawiki Lucene Search Extension. An interesting feature of the Lucene Search Extension, which can be perhaps exploited in the future, is its ability of to perform incremental updates of the index.

[1] Single threaded execution on Intel Xeon E5220, 15,000 rpm hard disk. The values reported exclude time required to map entities to Wikipedia article titles.

7.5. Applicability to Other Languages

All design and evaluation in this thesis was done with English as the target language. This followed from the requirements of the SCM and THD algorithms. Resource wise, the former relies on the free availability of a thesaurus, and the latter on the free availability of an encyclopedic resource. Also, the THD algorithm requires the availability of third party language processing tools: tokenizer and POS tagger. Last but not least, the core task in developing the THD algorithm was devising the grammar for the language used in the selected encyclopedic resource. It is clear that the author needs to have a good command of this language, in order to be able to devise a formal hypernym extraction grammar. This latter requirement effectively narrowed the choice of languages to Czech and English. Although there is a Czech version of Wikipedia, and a Czech version of WordNet (EuroWordNet), these resources have lower coverage than their English counterparts and they are not free.[2] Notwithstanding the availability of language processing tools for Czech, English was the natural choice.

7.5.1. SCM and THD

The circumstances of our choosing English for SCM and THD, as discussed above, hint at possibilities for extending these tools for other languages. Interestingly, for named entities, no changes at all may be required. The reason is that once the named entities are extracted from the input text, they tend to be language independent. Wikipedia also contains redirects from different spelling variants of the named entity. For example, English Wikipedia contains a redirect from Czech "Londýn" to "London". Mapping class names from a particular language to English WordNet is straightforward and can be automated.

Of course, using Wikipedia of the particular language has its benefits, even for named entities. Local versions are smaller, but they are not subsets of English Wikipedia. Many named entities of local importance not present in English Wikipedia are covered. However, use of non-English Wikipedia with THD requires a design of the extraction grammar for the particular language. While this requires only change of one file, the application needs to be recompiled with language processing tools (GATE PRs) for the particular language.

Perhaps the biggest issue is with the SCM algorithm, which consumes either directly words from the input text, or the result of THD. One possible solution is the use of EuroWordNet for the particular language, if available, another solution is to translate the input words.

7.5.2. BOA

One of the motives for designing the BOA algorithm was solving the language dependency of the SCM and THD algorithms. BOA core is a statistical NLP algorithm. We assume BOA to be applicable with no changes to any language from the Indo-European language family provided that a Wikipedia for this language exists. Using BOA for a new language generally requires only creating a Lucene index with the modified Lucene-search Mediawiki extension from the Wikipedia XML dump as described in Appendix B.7). It is also highly recommended to use stop-word list for the particular language. Given the relatively small improvement in performance from the use of WordNet, it is questionable whether the effort required to replace the English WordNet with EuroWordNet, if at all available for the given target language, is worthwhile.

[2] `http://www.illc.uva.nl/EuroWordNet/finalresults-ewn.html`

7.6. Additional Contribution

The development of the SCM and BOA algorithms resulted in several minor contributions of standalone applicability:

- Entity resolution component introduced in Subs. 1.2.1 is the only unsupervised algorithm known to the author which runs in near-real time against on-line Wikipedia.
- JAPE grammar designed as part of THD in Subs. 1.3.3 was one of the three used to construct the lexico-syntactic baseline in the 2011 paper [LLM11].
- New WordNet similarity based term-weighting function is proposed in Subs. 2.3.3. This term-weight vector was experimentally shown to improve classification results on the WordSim353 dataset.
- A framework for handling multiple term-weighting functions is proposed in Subs. 2.3.4.
- A new aggregator operator was proposed in two variants in Subs 2.3.5 for aggregating multiple term-weight vectors in situations, when some of the term-weight vectors may have missing entries.
- The official Lucene index created by the Mediawiki Lucene Search extension was found to provide sufficient information for our BOA algorithm, after a few minor changes to the indexing module as documented in Subs. 2.6.6. According to a preliminary analysis, it could be used as a knowledge source for other Wikipedia-based WSC algorithms, in particular for WLM and perhaps for ESA. We believe that using a standard data structure is preferable to a custom-designed solution.
- A general genetic algorithm which searches for best values of configuration parameters for a machine-learning algorithm is covered by Sec. 2.7.
- Survey of the newly established field of Wikipedia-based word similarity computation is given in Sec. 3.3.
- Chapter 4 presents a comprehensive survey of WordNet similarity measures and their implementations.
- Qualitative and quantitative evaluation of Wikipedia as a source of text for discovering hypernyms with Hearst patterns is presented in Chapter 5.
- Designed new dataset for entity classification (Czech Traveler dataset). This dataset, introduced in Subs. 6.1.3, contains 143 entities with inter-annotator agreement to nine classes. The dataset is listed in full with ground-truth and automatically established WordNet and Wikipedia mappings in Appendix D.
- Performed evaluation of Jiang and Conrath, Lin, Pirro and Seco, and Resnik WordNet similarity measures on the WordSim353 dataset. In addition to regular, corpus-based information content computation for Resnik and Lin, all the measures were evaluated also with intrinsic information content. The impact of sense selection (most frequent sense vs maximum similarity) was also assessed. The results are presented in Subs. 6.3.2.

We hope that if not directly the SCM or BOA algorithms, at least some of these additional results may provide an inspiration for future research.

Bibliography

[AAH+09] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalová, Marius Paşca, and Aitor Soroa, *A study on similarity and relatedness using distributional and WordNet-based approaches*, Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Stroudsburg, PA, USA), NAACL '09, Association for Computational Linguistics, 2009, pp. 19–27.

[Agi07] Eneko Agirre, *Word sense disambiguation: Algorithms and applications*, Springer-Verlag, 2007.

[AM02] Enrique Alfonseca and Suresh Manandhar, *An unsupervised method for general named entity recognition and automated concept discovery*, Proceedings of the 1st International Conference on General WordNet (India), 2002.

[ARG95] Eneko Agirre, German Rigau, and Pau Gargallo, *A proposal for word sense disambiguation using conceptual distance*, Proceedings of the Recent Advances in Natural Language Processing, RANLP '95, 1995.

[Ash05] Daniel Ashlock, *Evolutionary computation for modeling and optimization*, Springer, 2005.

[BBEF04] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, and D.A. Forsyth, *Who's in the picture?*, Neural Information Processing Systems Conference, 2004.

[BC07] Sebastian Blohm and Philipp Cimiano, *Using the web to reduce data sparseness in pattern-based information extraction*, Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases (Berlin, Heidelberg), PKDD 2007, Springer-Verlag, 2007, pp. 18–29.

[BDMP06] Holger Bast, Georges Dupret, Debapriyo Majumdar, and Benjamin Piwowarski, *Discovering a term taxonomy from term similarities using principal component analysis*, Proceedings of the 2005 joint international conference on Semantics, Web and Mining (Berlin, Heidelberg), EWMF'05/KDO'05, Springer-Verlag, 2006, pp. 103–120.

[BH06] Alexander Budanitsky and Graeme Hirst, *Evaluating WordNet-based measures of lexical semantic relatedness*, Computational Linguistics **32** (2006), no. 1, 13–47.

[BJ07] Ron Bekkerman and Jiwoon Jeon, *Multi-modal clustering for multimedia collections*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR '07, 2007.

[BKV03] Núria Bel, Cornelis H. A. Koster, and Marta Villegas, *Cross-lingual text categorization*, Proceedings of the 7th European Conference on Research and Advancement Technology for Digital Libraries (Traugott Koch and Ingeborg Sølvberg, eds.), ECDL '03, vol. 2769, Springer-Verlag, 2003, pp. 126–139.

[BP03] Satanjeev Banerjee and Ted Pedersen, *Extended gloss overlaps as a measure of semantic relatedness*, Proceedings of the 18th international joint conference on Artificial intelligence (San Francisco, CA, USA), IJCAI'03, Morgan Kaufmann Publishers Inc., 2003, pp. 805–810.

[BP06] Razvan Bunescu and Marius Pasca, *Using encyclopedic knowledge for named entity disambiguation*, Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (Trento, Italy), EACL '06, Association for Computational Linguistics, 2006, pp. 9–16.

[CKN+08] Krishna Chandramouli, Tomáš Kliegr, Jan Nemrava, Vojtěch Svátek, and Ebroul Isquierdo, *Query refinement and user relevance feedback for contextualized image retrieval*, Proceedings of 5th International Conference on Visual Information Engineering (Xi'en, China), VIE' 08, IET, 2008, pp. 453 – 458.

[CMB+12] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, Wim Peters, and et al, *Developing language processing components with GATE version 7 (a user guide)*, 2012.

[CMBT02] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan, *Gate: an architecture for development of robust hlt applications*, Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '02, Association for Computational Linguistics, 2002, pp. 168–175.

[CMT00] Hamish Cunningham, Diana Maynard, and Valentin Tablan, *JAPE - a Java Annotation Patterns Engine (Second edition), Department of Computer Science, University of Sheffield, 2000*, Tech. report, 2000, Technical Report.

[CST00] Nello Cristianini and John Shawe-Taylor, *An introduction to support vector machines : and other kernel-based learning methods*, 1st ed., Cambridge University Press, March 2000.

[Cuc07] Silviu Cucerzan, *Large-scale named entity disambiguation based on Wikipedia data*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (Prague, Czech Republic), EMNLP-CoNLL '07, Association for Computational Linguistics, June 2007, pp. 708–716.

[CV05a] Philipp Cimiano and Johanna Völker, *Text2onto: a framework for ontology learning and data-driven change discovery*, Proceedings of the 10th international conference on Natural Language Processing and Information Systems (Berlin, Heidelberg), NLDB'05, Springer-Verlag, 2005, pp. 227–238.

[CV05b] Philipp Cimiano and Johanna Völker, *Towards large-scale, open-domain and ontology-based named entity classification*, Proceedings of Recent Advances in Natural Language Processing, RANLP'05, 2005, pp. 166–172.

[CV07] Rudi L. Cilibrasi and Paul M. B. Vitanyi, *The Google similarity distance*, IEEE Transactions on Knowledge and Data Engineering **19** (2007), 370–383.

[Dee88] Scott Deerwester, *Improving information retrieval with Latent Semantic Indexing*, Proceedings of the 51st ASIS Annual Meeting (Atlanta, Georgia) (Christine L. Borgman and Edward Y. H. Pai, eds.), ASIS '88, vol. 25, American Society for Information Science, October 1988.

[DM05] Hal Daumé, III and Daniel Marcu, *Learning as search optimization: approximate large margin methods for structured prediction*, Proceedings of the 22nd international conference on Machine learning (New York, NY, USA), ICML '05, ACM, 2005, pp. 169–176.

[DM07] Koen Deschacht and Marie-Francine Moens, *Text analysis for automatic image annotation*, Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (Prague, Czech Republic), ACL'05, Association of Computational Linguistics, June 2007, pp. 1000–1007.

[Eva03] Richard Evans, *A framework for named entity recognition in the open domain*, Proceedings of the Recent Advances in Natural Language Processing, RANLP'03, 2003, pp. 137–144.

[Fel98] Christiane Fellbaum (ed.), *WordNet: An electronic lexical database (language, speech, and communication)*, illustrated edition ed., The MIT Press, May 1998.

[FGM+02] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin, *Placing search in context: the concept revisited*, ACM Transactions on Information Systems **20** (2002), no. 1, 116–131.

[FH02] Michael Fleischman and Eduard Hovy, *Fine grained classification of named entities*, Proceedings of the 19th international conference on Computational linguistics (Morristown, NJ, USA), Association for Computational Linguistics, 2002, pp. 1–7.

[FK83] Winthrop Nelson Francis and Jiří Kučera, *Frequency analysis of English usage: Lexicon and grammar*, Houghton Mifflin, 1983.

[Fri10] Craig Friedman, *Utility-based learning from data*, Chapman and Hall/CRC Machine Learning and Pattern Recognition, CRC Press, Hoboken, 2010.

[FS06] Ronen Feldman and James Sanger, *The text mining handbook: Advanced approaches in analyzing unstructured data*, Cambridge University Press, December 2006.

[FS10] Samuel Fernando and Mark Stevenson, *Aligning WordNet synsets and Wikipedia articles*, Proceedings of the Collaboratively-Built Knowledge Sources and Artificial Intelligence Workshop at 22nd international conference on Machine learning, Association for Computational Linguistics, 2010.

[FW86] Philip J. Fleming and John J. Wallace, *How not to lie with statistics: the correct way to summarize benchmark results*, Commun. ACM **29** (1986), 218–221.

[GAS99] Theo Gevers, Frank Aldersho, and Arnold W.M. Smeulders, *Classification of images on the internet by visual and textual information*, Proceedings of SPIE Conference on Internet Imaging, December 1999, pp. 16–27.

[GAS11] Thomas Gottron, Maik Anderka, and Benno Stein, *Insights into explicit semantic analysis*, Proceedings of the 20th ACM international conference on Information and knowledge management (New York, NY, USA), CIKM '11, ACM, 2011, pp. 1961–1964.

[GLM05] Alexander Genkin, David D. Lewis, and David Madigan, *Sparse logistic regression for text categorization*, Project Report, DIMACS Working Group on Monitoring Message Streams.

[GM07] Evgeniy Gabrilovich and Shaul Markovitch, *Computing semantic relatedness using Wikipedia-based explicit semantic analysis*, Proceedings of the 20th international joint conference on Artifical intelligence (San Francisco, CA, USA), IJCAI'07, Morgan Kaufmann Publishers Inc., 2007, pp. 1606–1611.

[Har92] Donna Harman, *The DARPA TIPSTER project*, SIGIR Forum **26** (1992), 26–28.

[Hea92] Marti A. Hearst, *Automatic acquisition of hyponyms from large text corpora*, Proceedings of the 14th conference on Computational linguistics - Volume 2 (Stroudsburg, PA, USA), COLING '92, Association for Computational Linguistics, 1992, pp. 539–545.

[Jac07] Henning Jacobs, *Explicit semantic analysis (ESA) using Wikipedia*, 2007, Retrieved June 7, 2011 from http://www.srcco.de/v/wikipedia-esa.

[Jar03] Mario Jarmasz, *Roget's thesaurus as a lexical resource for natural language processing*, Tech. report, University of Ottowa, 2003.

[JC97] Jay J. Jiang and David W. Conrath, *Semantic similarity based on corpus statistics and lexical taxonomy*, Proceedings of the International Conference on Research in Computational Linguistics, 1997, pp. 19–33.

[Joa97] Thorsten Joachims, *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*, Proceedings of the Fourteenth International Conference on Machine Learning (San Francisco, CA, USA), ICML '97, Morgan Kaufmann Publishers Inc., 1997, pp. 143–151.

[KCN+08a] Tomáš Kliegr, Krishna Chandramouli, Jan Nemrava, Vojtěch Svátek, and Ebroul Izquierdo, *Combining image captions and visual analysis for image concept classification*, Proceedings of the 9th International Workshop on Multimedia Data Mining: held in conjunction with the ACM SIGKDD 2008 (New York, NY, USA), MDM '08, ACM, 2008, pp. 8–17.

[KCN+08b] Tomáš Kliegr, Krishna Chandramouli, Jan Nemrava, Vojtěch Svátek, and Ebroul Izquierdo, *Wikipedia as the premiere source for targeted hypernym discovery*, Proceedings of the Wiki's, Blogs and Bookmarking tools - Mining the Web 2.0 Workshop at ECML'08, 2008.

[Kli10] Tomáš Kliegr, *Entity classification by bag of Wikipedia articles*, Proceedings of the 3rd workshop on Ph.D. students in information and knowledge management (New York, NY, USA), PIKM '10, ACM, 2010, pp. 67–74.

[KR00] Adam Kilgarriff and Joseph Rosenzweig, *Framework and results for English SENSEVAL*, Special Issue on SENSEVAL. Computers and the Humanties (2000), 15–48.

[KT07] Jun'ichi Kazama and Kentaro Torisawa, *Exploiting Wikipedia as external knowledge for named entity recognition*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL'07, 2007, pp. 698–707.

[LC98] Claudia Leacock and Martin Chodorow, *Combining local context with WordNet similarity for word sense identification*, WordNet: A Lexical Reference System and its Application, 1998.

[Les86] Michael Lesk, *Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone*, Proceedings of the 5th annual international conference on systems documentation (New York, NY, USA), SIGDOC '86, ACM, 1986, pp. 24–26.

[Lin98] Dekang Lin, *An information-theoretic definition of similarity*, In Proceedings of the 15th International Conference on Machine Learning, Morgan Kaufmann, 1998, pp. 296–304.

[LLM11] Berenike Litz, Hagen Langer, and Rainer Malaka, *Sequential supervised learning for hypernym discovery from Wikipedia*, Knowledge Discovery, Knowlege Engineering and Knowledge Management (Ana Fred, Jan L. G. Dietz, Kecheng Liu, and Joaquim Filipe, eds.), Communications in Computer and Information Science, vol. 128, Springer-Verlag, Berlin Heidelberg, 2011, pp. 68–80.

[Mil09] David Milne, *An open-source toolkit for mining Wikipedia*, Proceedings of the New Zealand Computer Science Research Student Conference, 2009.

[MLTB93] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker, *A semantic concordance*, Proceedings of the workshop on Human Language Technology (Stroudsburg, PA, USA), HLT '93, Association for Computational Linguistics, 1993, pp. 303–308.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze, *Introduction to information retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[MS92] Christopher D. Manning and Hinrich Schultze, *Foundations of statistical natural language processing*, MIT Press, 1992.

[Mut99] James E. De Muth, *Basic statistics and pharmaceutical statistical applications, 2nd edition*, Journal of the Royal Statistical Society: Series A (1999).

[MW08] David Milne and Ian H. Witten, *An effective, low-cost measure of semantic relatedness obtained from Wikipedia links*, Proceedings of the first AAAI Workshop on Wikipedia and Artificial Intelligence, 2008, pp. 25–30.

[NS07] David Nadeau and Satoshi Sekine, *A survey of named entity recognition and classification*, Linguisticae Investigationes **30** (2007), no. 1, 3–26.

[OSdCI11] Jesus Oliva, Jose Ignacio Serrano, María Dolores del Castillo, and Ángel Iglesias, *Symss: A syntax-based measure for short-text semantic similarity*, Data & Knowledge Engineering **70** (2011), no. 4, 390–405.

[PBP05] Ted Pedersen, Satanjeev Banerjee, and Siddharth Patwardhan, *Maximizing semantic relatedness to perform word sense disambiguation*, Research Report **25** (2005), 2005–25.

[PDKM09] Judita Preiss, Jon Dehdari, Josh King, and Dennis Mehay, *Refining the most frequent sense baseline*, Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (Stroudsburg, PA, USA), DEW '09, Association for Computational Linguistics, 2009, pp. 10–18.

[PS08] Giuseppe Pirró and Nuno Seco, *Design, implementation and evaluation of a new semantic similarity metric combining features and intrinsic information content*, Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part II on On the Move to Meaningful Internet Systems (Berlin, Heidelberg), OTM '08, Springer-Verlag, 2008, pp. 1271–1288.

[RB89] Roy Rada and Ellen Bicknell, *Ranking documents with a thesaurus*, Journal of the American Society for Information Science (1989), 304–310.

[Res95] Philip Resnik, *Using information content to evaluate semantic similarity in a taxonomy*, Proceedings of the 14th international joint conference on Artificial intelligence - Volume 1 (San Francisco, CA, USA), IJCAI'95, Morgan Kaufmann Publishers Inc., 1995, pp. 448–453.

[RM95] Lance A. Ramshaw and Mitchell P. Marcus, *Text chunking using transformation-based learning*, Third Workshop on Very Large Corpora (Somerset, New Jersey) (David Yarovsky and Kenneth Church, eds.), Association for Computational Linguistics, 1995, pp. 82–94.

[Roc71] J. Rocchio, *Relevance feedback in information retrieval*, The SMART Retrieval System, 1971, pp. 313–323.

[SB88] Gerard Salton and Christopher Buckley, *Term-weighting approaches in automatic text retrieval*, Information Processing and Management, Pergamon Press, 1988, pp. 513–523.

[SJN05] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng, *Learning syntactic patterns for automatic hypernym discovery*, Advances in Neural Information Processing Systems (Cambridge, MA), no. 17, MIT Press, 2005, pp. 1297–1304.

[SJN06] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng, *Semantic taxonomy induction from heterogenous evidence*, Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (Stroudsburg, PA, USA), ACL-44, Association for Computational Linguistics, 2006, pp. 801–808.

[SKA09] Mohammed M. Sakre, Mohammed M. Kouta, and Ali M. N. Allam, *Weighting query terms using Wordnet ontology*, International Journal of Computer Science and Network Security **9** (2009), 349–358.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum, *Yago: a core of semantic knowledge*, Proceedings of the 16th international conference on World Wide Web (New York, NY, USA), WWW '07, ACM, 2007, pp. 697–706.

[SNK99] Shin'ichi Satoh, Yuichi Nakamura, and Takeo Kanade, *Name-it: Naming and detecting faces in news videos*, IEEE MultiMedia **6** (1999), no. 1, 22–35.

[SP06] Michael Strube and Simone Paolo Ponzetto, *WikiRelate! computing semantic relatedness using Wikipedia*, Proceedings of the 21st national conference on Artificial intelligence - Volume 2, AAAI'06, AAAI Press, 2006, pp. 1419–1424.

[SSN02] Satoshi Sekine, Kiyoshi Sudo, and Chikashi Nobata, *Extended named entity hierarchy*, Proceedings of 3^{rd} International Conference on Language Resources and Evaluation (Canary Islands, Spain) (M. Gonzáles Rodríguez and C. Paz Suárez Araujo, eds.), LREC'02, May 2002, pp. 1818–1824.

[ST93] Daniel D. K. Sleator and Davy Temperley, *Parsing English with a link grammar*, Proceedings of the 3rd International Workshop on Parsing Technologies, 1993.

[Tuc89] Allan Tucker, *A unified introduction to linear algebra: Models, methods and theory*, Maxwell Macmillan, 1989.

[Tve77] Amos Tversky, *Features of similarity*, Psychological Review **84** (1977), 327–352.

[WD08] Pu Wang and Carlotta Domeniconi, *Building semantic kernels for text classification using Wikipedia*, Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), KDD '08, ACM, 2008, pp. 713–721.

[Wes00] Thijs Westerveld, *Image retrieval: Content versus context*, Proceedings of RIAO 2000 Conference on Content-Based Multimedia Information Access, 2000, pp. 276–284.

[WHZ+07] Pu Wang, Jian Hu, Hua-Jun Zeng, Lijun Chen, and Zheng Chen, *Improving text classification by using encyclopedia knowledge*, Proceedings of the 7th IEEE International Conference on Data Mining (Washington, DC, USA), IEEE Computer Society, 2007, pp. 332–341.

[WMS00] Peng Wu, Bangalore S. Manjunath, and Hyundoo D. Shin, *Dimensionality reduction for image retrieval*, Proceedings of International Conference on Image Processing, vol. 3, IEEE, 2000, pp. 726 – 729.

[WP94] Zhibiao Wu and Martha Palmer, *Verbs semantics and lexical selection*, Proceedings of the 32nd annual meeting on Association for Computational Linguistics (Stroudsburg, PA, USA), ACL '94, Association for Computational Linguistics, 1994, pp. 133–138.

[WT91] William E. Winkler and Yves Thibaudeau, *An application of the Fellegi-Sunter model of record linkage to the 1990 U.S. decennial census*, Tech. report, U.S. Bureau of the Census, Washington, D.C., 1991, Statistical Research Report Series RR91/09.

[ZAAS05] Ana Zelaia, Inaki Alegria, Olatz Arregi, and Basilio Sierra, *Analyzing the effect of dimensionality reduction in document categorization for Basque*, Archives of Control Science (2005).

List of Acronyms

BOA Bag of Articles (a contributed algorithm)

CC Conjunction POS tag

CD Cardinal Number POS tag

ESA Explicit Semantic Analysis algorithm

GATE General Architecture for Text Engineering (software bundle for NLP)

IC Information Content

IDF Inverse Document Frequency

IN Preposition or subordinating conjunction

JAPE Java Annotation Pattern Engine (regular expressions over linguistic annotations)

JCn Jiang and Conrath WordNet similarity measure

JJ Adjective POS tag

JWNL Java WordNet Library

JWSL Java WordNet Similarity Library

MFS Most Frequent Sense

NE Named Entity

NER Named Entity Recognition

NLP Natural Language Processing

NN Noun POS tag

NNP Singular Proper Noun POS tagg

NNS Plural Proper Noun POS tag

NP Noun Phrase

POS Part of Speech

PR Processing Resource (GATE module)

P&S Pirro and Seco WordNet similarity measure

RB Adverb POS tag

SCM Semantic Concept Mapping (a contributed algorithm)

SSM Synset Similarity Maximization

TF Term Frequency

THD Targeted Hyperym Discovery (a generic task and contributed algorithm)

VBN Verb past participle POS tag

WLM Wikipedia Link Measure algorithm

WSC Word Similarity Computation

WSD Word Sense Disambiguation

Appendices

A. JAPE grammar

This Appendix lists the complete JAPE grammar used by our THD implementation.

Listing 14 shows several rules that create support annotations and features, which are used by the Hearst pattern grammars: rule `TransferSplitAnnotationToTokenFeature` serves for transferring the Split annotation created by Sentence Splitter to true value of the `split` feature on the overlapping `Token` annotation, rule `TokenDoesNotOverlapWithSplit` ensures that if the Token annotation does not overlap with the `Split` annotation, the value of the `split` feature is false. The `MarkStartOfText` rule marks the first token with the `bof` feature. Listing 15 contains macros referenced from the Hearst pattern grammars.

Listing 16 shows the experimental Hearst pattern grammar, which used the `Token bof` feature instead of the Highlight annotation. Listing 17 shows the main Hearst pattern grammar.

Algorithm 14 Preparation phases

```
Phase: TransferSplitAnnotationToTokenFeature
Input: Token Split
Options: control=appelt
Rule: TokenOverlapsWithSplit
Priority:1000
({Split}):split
 ⟹
{
//get the annotation set
gate.AnnotationSet annSet = (gate.AnnotationSet) bindings.get("split");
//get the only annotation from the set
gate.Annotation splitAnn = (gate.Annotation)annSet.iterator().next();
//get set of tokens overlapping with current annotation
gate.AnnotationSet tokenAS = inputAS.get("Token",
splitAnn.getStartNode().getOffset(),
splitAnn.getEndNode().getOffset());
//if this check is not there an error occurs
if (tokenAS.isEmpty())
return;
//add nonsplit feature to Token annotation
tokenAS.iterator().next().getFeatures().put("split", "true");
}

Rule: TokenDoesNotOverlapWithSplit
({Token}):nonSplitToken
 ⟹
:nonSplitToken{
nonSplitTokenAnnots.iterator().next().getFeatures().put("split", "false");
}

Phase: BOFmark
Input: Token
Options: control=once

Rule: MarkStartOfText

({Token}):firstToken
 ⟹
:firstToken{
firstTokenAnnots.iterator().next().getFeatures().put("bof", "true");
}
```

Algorithm 15 Phase SimpleHearst – Macros

```
Phase: SimpleHearst
Input: Token Split Highlight
Options: control = appelt

Macro: LHSHearstBody
(
({Token.split=="false"})?
({Token.category == "CD"})?
({Token.category == "JJ"})?
({Token.category == "JJ"})?
({Token.category == "NNP"})?
({Token.category == "NNP"})?
({Token.category == "VBN"})?
({Token.category == "JJ"})?
({Token.category == "JJ"})?
({Token.category == "NN"})?
({Token.category == "NN"})?
({Token.category == "NN"})?
)

Macro: Head
(
({Token.category == "NN"}|{Token.category == "NNP"}|{Token.category == "NNS"})
)

Macro: NameOf
(
({Token.string == "name"})
({Token.string == "of"})
)
```

Algorithm 16 Phase SimpleHearst – HearstRule_ArticleStart

```
Rule: HearstRule_ArticleStart
Priority: 30
(
{Token.string == "is",Token.bof =="true"}
|{Token.string == "are",Token.bof =="true"}
|{Token.string == "were",Token.bof =="true"}
|{Token.string == "was",Token.bof =="true"}
)
({Token.string == "a"}|{Token.string == "an"}|{Token.string == "the"}):hearstArticle
(NameOf)?
(LHSHearstBody)
(Head)
:hearstPattern
 ⟹
//there is no java necessary for article start match, since there is no highlight annotation.
:hearstArticle.harticle =
{
  kind = "isApattern",
  rule = "HearstRule_ArticleStart"
},
:hearstPattern.hearst =
{
  kind = "isApattern",
  rule = "HearstRule_ArticleStart",
  type = "article_start"
}
```

Algorithm 17 Phase SimpleHearst – Rule HearstRule_Normal

```
Rule: HearstRule_Normal
Priority: 50
({Highlight}):highlight
({Token.split=="false"})*
({Token.string == "is"}|{Token.string == "are"}|{Token.string == "were"}|{Token.string
== "was"})
({Token.string == "a"}|{Token.string == "an"}|{Token.string == "the"}):hearstArticle
(NameOf)?
(LHSHearstBody)
(Head)
:hearstPattern
 ⟹
:hearstArticle.harticle = {kind = "isApattern", rule = "HearstRule_Normal"},
:hearstPattern {
//get features on the highlighted query
AnnotationSet highlightAnnots = (AnnotationSet)bindings.get("highlight");
Annotation theHighlightAnnot = (Annotation)highlightAnnots.iterator().next();
String diacriticsMatches = (String) theHighlightAnnot.getFeatures().get("DiacriticsMatches");

String caseMatches = (String) theHighlightAnnot.getFeatures().get("CaseMatches");
String type = (String) theHighlightAnnot.getFeatures().get("Type");
String full = (String) theHighlightAnnot.getFeatures().get("Full");
//get the hearstpattern annotation,
AnnotationSet hearstPatternAnnotations = (AnnotationSet)bindings.get("hearstPattern");

//create new feature map
gate.FeatureMap features = Factory.newFeatureMap();
//Annotation hearstPatternAnnot = (Annotation)hearstPatternAnnotations.iterator().next();

//transfer features from highlight to the hearstpattern annotation
features.put("DiacriticsMatches",diacriticsMatches);
features.put("CaseMatches",caseMatches);
features.put("Type",type);
//create features of the hearst match
features.put("kind","isApattern");
features.put("rule","HearstRule_Normal");
//append the new annotation
outputAS.add(hearstPatternAnnotations.firstNode(),
hearstPatternAnnotations.lastNode(), "hearst", features);
}
```

B. BOA and SCM Implementation

This appendix lists configuration options for our classifier program `WikiIndex.jar`. While it is primarily an implementation of BOA, it also allows to use WordNet similarity measures, which are at the core of the SCM algorithm. This software was used to perform all experiments in Chapter 6.

It should be noted that since it does not support THD and entity extraction, these were performed by our older standalone SCM implementation described in Sec. 1.5. The entities were then provided to `WikiIndex.jar` already mapped to WordNet entries.

The result of the program is a Spearman rank correlation coefficient or accuracy, depending on the experiment type.

The application is available at `http://nb.vse.cz/~klit01`.

B.1. Experiment Types

The implementation allows to use several experiment types. The root element of the configuration file defines the name of the `ExperimentConfig` class, which will be used to parse the configuration file (e.g. `evaluation.BOAExperimentConfig` in config file from Sec. B.4). The name of the class, which will carry out the experiment is the name of the root element without the "Config" suffix, e.g. `evaluation.BOAExperiment`.

The following experiment types are currently defined:

- `BOAExperiment` is intended for classification experiments with the BOA algorithm. The ground-truth is given in the form of a correct class for unlabeled instance. The return value of this experiment is accuracy, $accuracy = correct/all$, where *correct* is a count of testing instances for which the classification result is the same as the entry in the ground-truth file, and *all* is the total number of testing instances involved.
- `SCMExperiment` is intended for classification experiments as described above with SCM.
- `WordSim353` is intended for experiments on the WordSim353 collection, where the BOA classifier is used to compute a similarity between two input instances. The ground-truth is given in the form of similarity values given for the input word pairs. The configuration file for this type of experiment does not contain the `EntityClassifierTestingConfig` element. The input dataset consists of pairs of terms, each pair being assigned a similarity score. These similarity scores can be translated to a total order of the pairs. The result of the experiment is a value of the Spearman Rank correlation coefficient.
- `WordSim353Wordnet` is intended for experiments on WordSim353 with SCM.

Individual experiments are executed using the `evaluation.ExperimentRunner` class with one argument – the path to the XML configuration file:

```
java -jar WikiIndex.jar ExperimentConfig.xml
```

B.2. Experiment Configuration File

Experiment configuration file is an XML file with a number of parameters, which can be divided into several groups: global, search-related, modality and term-weighting functions.

The names of parameters often include a full path to the class, which uses the parameter. For the sake of brevity, the configuration files were simplified by abridging these rather lengthy parameter names. Parameters common for both SCM and BOA are listed in Sec. B.2.1. Settings specific to BOA are listed in Subs. B.2.2, and settings specific to SCM in Subs. B.2.2.

In addition to the description of the parameters, we give two examples. An example for BOA experiment on a classification task is given in Appendix B.4. An example for SCM on a WordSim353 task is given in Appendix B.5.

B.2.1. Common Parameters

Tables B.3-B.9 give an overview of the majority of available parameters, which are common for BOA and SCM. Parameters `additionalDebugFilesDir`, `debugLogPath` and `protocolPath` can be missing. In that case, directory where the experiment file resides is the BASE and `protocolPath` is set to `BASE/protocol.xml`, `debugLogPath` to `BASE/debug_details` directory and `debugLogPath` to `BASE/debug.log`.

Global parameters		
experimentClass	enum	either `evaluation.WordSim353` (BOA), `evaluation.WordSim353WordNet` (SCM) for computation on WordSim353 or similar dataset with Spearman correlation as result or `evaluation.BOAExperiment` (BOA), `evaluation.SCMExperiment` (SCM) for classification task with accuracy as the result
Global path parameters		
loggerLevel	enum	logging granularity (`ERROR`, `INFO`, `DEBUG`, `TRACE`)
consoleLoggingEnabled	booelan	if set to true, log messages will be sent also to standard output
protocolPath	string	detailed result of the experiment run is saved into this file
debugLogPath	string	logging messages will be saved to this file
additionalDebugFilesDir	string	if loggerLevel is set to `DEBUG`, one .csv file per vector similarity computation will be saved to this directory

Table B.1.: Global technical parameters.

B.2.2. BOA Specific Parameters

Global parameters		
similarityFunction	enum	either `dotProduct` or `cosineSim`
wiki_linksDir	string	path to Lucene wiki.links directory
wiki_mainIndexDir	string	path to Lucene wiki directory
wiki_useRAMDir	boolean	Lucene index will be loaded to RAM Directory. Note that for English Wikipedia index this requires an excessive amount of RAM.

Table B.2.: Global Parameters.

EntityClassifierTrainingConfig – Training parameters		
entity-NamesAreWikipedia-ArticleTitles	boolean	the input strings are considered as titles of Wikipedia articles (*true*), as noun phrases (*false*)
maxTermVectorLength	integer	maximum length of term vectors return by σ, see 2.3.2
skipUnresolvedTitles	boolean	If the entity cannot be mapped to Wikipedia, the training quits if set to `false`, if set to `true` the training continues with the unmapped training class being omitted.
stopWordListPath	string	Empty if stop word list is not to be used. See 2.6.3

Table B.3.: Training Parameters

EntityClassifierSearchConfig		
disambiguationCutoff	integer	number of best matching articles to retrieve per query. Default is 1, greater values leave space for disambiguation
searchBackupPages	integer	number of search results to ask the search engine in addition to `disambiguationCutoff`. The excess hits are used if the system fails to parse some of the top `disambiguationCutoff` results.
searchServiceURL	string	location of the `routerAPISeach.php` script wrapping the Wikipedia Lucene Extension search
searchType	enum	possible values are `rawexplain`, `search`. Refer to Lucene search documentation.
pathToFileWithLucene-ArticleKeys	string	entity names (first column) are replaced with Wikipedia article titles (second column).
searchProvider	enum	`Service` – search by service at `searchServiceURL`, `KeysFromFile` use keys at `pathToFileWithLuceneArticleKeys`

Table B.4.: Search Parameters

MultipleWeightSparseTermVectorType		
TV_Char	enum	always value `TF`
TV_Scope	enum	always value `entity`
TV_UseType	enum	value `training` or `testing`

Table B.5.: Basic Term Weighting Parameters

Basic Wordnet Config		
discardTermsNotIn-Wordnet	boolean	Refer to Subs. 2.6.3
TVChar_Wordnet_JWNL		
infoContentFileName	string	path to file with precomputed information content values (refer to Subs. 4.3.1)
jwnlinitPath	string	path to file with WordNet setup (options: file-based/memory-based); influences speed but not results (refer to Subs. 4.3.2)
TVChar_Wordnet_JWSL		
wordnetLucenefolder	string	path to the JWSL Lucene index directory

Table B.6.: Basic WordNet Config

Modalities: TV_Linkxxx Possible values:TV_LinkSimByCat, TV_LinkOut,TV_LinkIN		
crawlingDepth	integer	corresponds to L^m_{max} threshold in Eq. (2.20)
WeightingFactor	$\langle 0, 1\rangle$	corresponds to W_m in Eq. (2.19)
WeightFactor_level$x$	$\langle 0, 1\rangle$	corresponds to $W_{m,l}$ in Eq. (2.20). Must be set for $x = 0 \ldots crawlingDepth$.
maxLinksToFollow	integer	number of articles in level $n+1$ related to article a on level n to use (refer to Subs 2.6.2)
articleSelectionStrategy	enum	$firstn$ or $mostsim$ (refer to Subs. 2.6.2)

Table B.7.: Modality Config Parameters

Term weighting functions: TVChar_xxx where xxx $\in$ {TermFrequency, IDFentireWikipedia,IDFtrainingSet, Wordnet_Aggregate, Wordnet_JWNL, WordNet_JWSL }		
WeightFactor_level$x$	$\langle 0; 1\rangle$	corresponds to weight $W_{m,l,t}$ – refer to Eq. (2.23). Must be set for $x = 0 \ldots crawlingDepth$.

Table B.8.: Term Weighting Function Config Parameters

Additional settings for TVChar_Wordnet_xxx where xxx $\in$ {Wordnet_Aggregate, Wordnet_JWNL, WordNet_JWSL		
roundToZeroIf-UnderWordnetSim-Threshold_level$x$	float	corresponds to weight T^{low}_l in Eq. (2.109), this property must be present for $x = 0 \ldots crawlingDepth$
roundToOneIf-AboveWordnetSim-Threshold_level$x$	float	corresponds to weight T^{high}_l in Eq. (2.109), this property must be present for $x = 0 \ldots crawlingDepth$
Additional settings for TVChar_Wordnet_JWNL		
Wordnet_simMetric	enum	*shef.nlp.wordnet.similarity.{JCn,Lin}*
Additional settings for TVChar_Wordnet_JWSL		
Wordnet_simMetric	enum	{*Resnik, Jiang, Lin, Pirro and Seco*}
Additional settings for TVChar_Wordnet_Aggregate		
WordnetWeight	nested structure	list of configurations: TVChar_Wordnet_{JWNL,JWSL}.

Table B.9.: WordNet Term Weighting Function Specific Config Parameters

B.2.3. SCM Specific Parameters

SCM contains a subset of parameters available for BOA. There are only two SCM specific parameters:

- `JWSLMeasures` element lists a semicolon separated list of JWSL WordNet measures to be used,
- `JWNLMeasures` element lists a semicolon separated list of JWordnetSim measures to be used.

B.3. Parameter Estimation

Parameter estimation is executed by the following command:

```
java -jar WikiIndex.jar BOAConfig.xml GAConfig
```

The two arguments are:

- path to the XML file, which is a normal BOA configuration file as e.g. exemplified in Sec. B.4 (`BOAConfig.xml`),
- path to the Genetic Algorithm Configuration file (`GAConfig`), which contains the setting of the genetic algorithm and a list of parameters that should be subject of optimization.

B.3.1. GAConfig Configuration File

This file consists of two sections. The first section contains generic settings for the genetic algorithm, and the second part defines features that are subject to evolution. The syntax for entries in the first part is simple: the name of the parameter is followed by space and then by value.

The syntax for the second part is following:

```
parameter name,min value,max value,parameter type,context
```

The `context` is given by a regular expression. The name of the parameter is searched in the config file fragment matching the context and replaced by a new value.

An example of GAConfig file for a BOA classification experiment follows.

This example defines 3 features that do not depend on phase, 17 features for the training phase, and 9 features for the classification phase.

```
maxGenerations 50
populationSize 60
maxThreads 8
maxGensWithoutImprovement 5
mutationProb 0.2
experimentExecutionType separateJVM
executionCommandforJVMExecutionType java -jar -Xmx2000M ~/WikiIndex.jar
debugLogPath /home/tomas/code/WIKIENTITYCLAS/WikiIndex/experiments/test/GA.log

wikiindex.characteristic.TVChar_Wordnet_JWNL_discardTermsNotInWordnet, true;
    false, , enum, .*
wikiindex.config.EntityClassifierTrainingConfig_maxTermVectorLength, 10, 50,
    gaussianInteger, .*
evaluation.BOAExperimentConfig_similarityFunction, dotProduct;cosineSim, ,enum
    , .*

wikiindex.characteristic.TV_LinkOut_crawlingDepth, 0;1;2, , enum, TV_LinkOut
    .*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_WeightFactor_level0, 0, 1, float,
    TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_WeightFactor_level1, 0, 1, float,
    TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_WeightFactor_level2, 0, 1, float,
    TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_maxLinksToFollow, 1, 20, integer,
    TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
```

```
wikiindex.characteristic.TV_LinkOut_weightingFactor, 0, 1, float, TV_LinkOut
    .*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_articleSelectionStrategy, firstn;
    mostsimilar, ,enum, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TV_LinkOut_aggregationType,
    CustomAggregator_PreserveBasicWeight;WeightedGeometricAverage;
    CustomAggregator, ,enum, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig

wikiindex.characteristic.TVChar_TermFrequency_WeightFactor_level0, 0, 1, float
    , TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>
    .*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_TermFrequency_WeightFactor_level1, 0, 1, float
    , TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>
    .*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_TermFrequency_WeightFactor_level2, 0, 1, float
    , TV_LinkOut.*?</wikiindex.termvector.MultipleWeightSparseTermVectorType>
    .*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFentireWikipedia_WeightFactor_level0, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFentireWikipedia_WeightFactor_level1, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFentireWikipedia_WeightFactor_level2, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFtrainingSet_WeightFactor_level0, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFtrainingSet_WeightFactor_level1, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig
wikiindex.characteristic.TVChar_IDFtrainingSet_WeightFactor_level2, 0, 1,
    float, TV_LinkOut.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>.*?EntityClassifierTrainingConfig

wikiindex.characteristic.TV_LinkIN_crawlingDepth, 0;1, , enum, TV_LinkOut.*?</
    wikiindex.termvector.MultipleWeightSparseTermVectorType>.*?
    EntityClassifierTestingConfig
wikiindex.characteristic.TV_LinkIN_WeightFactor_level0, 0, 1, float,
    EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TV_LinkIN_WeightFactor_level1, 0, 1, float,
    EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
```

```
    MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TV_LinkIN_maxLinksToFollow, 1, 20, integer,
    EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TV_LinkIN_weightingFactor, 0, 1, float,
    EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TV_LinkIN_articleSelectionStrategy, firstn;
    mostsimilar, ,enum, EntityClassifierTestingConfig.*?TV_LinkIN.*?</
    wikiindex.termvector.MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TV_LinkIN_aggregationType,
    CustomAggregator_PreserveBasicWeight;WeightedGeometricAverage;
    CustomAggregator, ,enum, EntityClassifierTestingConfig.*?TV_LinkIN.*?</
    wikiindex.termvector.MultipleWeightSparseTermVectorType>

wikiindex.characteristic.TVChar_TermFrequency_WeightFactor_level0, 0, 1, float
    , EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>
wikiindex.characteristic.TVChar_TermFrequency_WeightFactor_level1, 0, 1, float
    , EntityClassifierTestingConfig.*?TV_LinkIN.*?</wikiindex.termvector.
    MultipleWeightSparseTermVectorType>
```

It should be noted that no integrity checks are performed to ensure that value changes within the provided bounds will generate a valid configuration file. In the listing provided above, changing

```
wikiindex.characteristic.TV_LinkOut_crawlingDepth, 0;1;2, , enum, TV_LinkOut
```

to

```
wikiindex.characteristic.TV_LinkOut_crawlingDepth, 0;1;2;3, , enum, TV_LinkOut
```

can result in an invalid configuration if the *crawlingDepth* feature is set to value 3 through mutation. In this case, the implementation will search for level 3 parameters in the `BOA Config` XML file (refer to Sec. B.4), such as:

```
<LinkOut_WeightFactor_level3>*</LinkOut_WeightFactor_level3>
```

If the corresponding parameters are not present, the program will finish with an error. However, the BOA config file can contain extra parameters. For example, setting *crawlingDepth* to 1 for the in-link modality will result in a valid configuration, the extra parameters for level 2 which may be present in the configuration file will be ignored. An example of such a parameter is

```
<LinkIN_WeightFactor_level2>*</LinkIN_WeightFactor_level2>
```

B.4. BOA Experiment Config Example

This appendix lists a BOA configuration file for the classification task used as an example in Subs. 2.5.

Note that some lines which are not needed by the example, but are required for various integrity and technical reasons, were omitted from the listing.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<evaluation.BOAExperimentConfig>
 <additionalDebugFilesDir>debug-details</additionalDebugFilesDir>
 <consoleLoggingEnabled>true</consoleLoggingEnabled>
 <debugLogPath>debug.log</debugLogPath>
 <experimentClass>evaluation.BOAExperiment</experimentClass>
 <experimentName>testExperiment</experimentName>
 <groundtruthFile>groundtruth.csv</groundtruthFile>
 <groundtruth_col>0</groundtruth_col>
 <loggerLevel>DEBUG</loggerLevel>
 <protocolPath>protocolf1.csv</protocolPath>
 <serializationPath>serializedClassifier</serializationPath>
 <test_col>0</test_col>
 <test_from>0</test_from>
 <test_to>0</test_to>
 <testingFile>testing.csv</testingFile>
 <train_entityname_col>0</train_entityname_col>
 <train_col>1</train_col>
 <train_from>0</train_from>
 <train_to>1</train_to>
 <trainingFile>training.csv</trainingFile>
 <similarityFunction>cosineSim</similarityFunction>
 <wiki_linksDir>wiki.links</wiki_linksDir>
 <wiki_mainIndexDir>wiki</wiki_mainIndexDir>
 <wikiindex.config.SerializationPolicyEnum>noserialize</wikiindex.config.
    SerializationPolicyEnum>
 <wikiindex.config.EntityClassifierTrainingConfig>
  <wikiindex.config.EntityClassifierTrainingConfig>
   <entityNamesAreWikipediaArticleTitles>true</
      entityNamesAreWikipediaArticleTitles>
   <maxTermVectorLength>10</maxTermVectorLength>
   <skipUnresolvedTitles>false</skipUnresolvedTitles>
   <stopWordListPath>stopwordlist.txt</stopWordListPath>
   <wikiindex.config.EntityClassifierSearchConfig>
    <wikiindex.config.EntityClassifierSearchConfig>
     <disambiguationCutoff>1</disambiguationCutoff>
     <searchBackupPages>10</searchBackupPages>
     <searchServiceURL>routerAPISeach.php</searchServiceURL>
     <searchType>search</searchType>
     <pathToFileWithLuceneArticleKeys>searchkeys.csv</
```

```
      pathToFileWithLuceneArticleKeys>
   <searchProvider>KeysFromFile</searchProvider>
  </wikiindex.config.EntityClassifierSearchConfig>
 </wikiindex.config.EntityClassifierSearchConfig>
 <wikiindex.termvector.MultipleWeightSparseTermVectorType>
  <wikiindex.termvector.MultipleWeightSparseTermVectorType>
   <wikiindex.characteristic.TV_Char>TF</wikiindex.characteristic.TV_Char>
   <wikiindex.characteristic.TV_Scope>entity</wikiindex.characteristic.
      TV_Scope>
   <wikiindex.characteristic.TV_UseType>training</wikiindex.characteristic.
      TV_UseType>
   <LinkType>
    <wikiindex.characteristic.TV_LinkOut>
     <LinkOut_WeightFactor_level0>0.5</LinkOut_WeightFactor_level0>
     <LinkOut_WeightFactor_level1>0.4</LinkOut_WeightFactor_level1>
     <LinkOut_WeightFactor_level2>0.1</LinkOut_WeightFactor_level2>
     <LinkOut_crawlingDepth>2</LinkOut_crawlingDepth>
     <LinkOut_maxLinksToFollow>3</LinkOut_maxLinksToFollow>
     <LinkOut_weightingFactor>0.4</LinkOut_weightingFactor>
     <LinkOut_articleSelectionStrategy>firstn</
        LinkOut_articleSelectionStrategy>
     <LinkOut_aggregationType>WeightedGeometricAverage</
        LinkOut_aggregationType>
    </wikiindex.characteristic.TV_LinkOut>
   </LinkType>
   <TermVectorChars>
    <wikiindex.characteristic.TVChar_TermFrequency>
     <TermFrequency_WeightFactor_level0>0.3</
        TermFrequency_WeightFactor_level0>
     <TermFrequency_WeightFactor_level1>0.4</
        TermFrequency_WeightFactor_level1>
     <TermFrequency_WeightFactor_level2>0.5</
        TermFrequency_WeightFactor_level2>
    </wikiindex.characteristic.TVChar_TermFrequency>
    <wikiindex.characteristic.TVChar_IDFentireWikipedia>
     <IDFentireWikipedia_WeightFactor_level0>0.7</
        IDFentireWikipedia_WeightFactor_level0>
     <IDFentireWikipedia_WeightFactor_level1>0.6</
        IDFentireWikipedia_WeightFactor_level1>
     <IDFentireWikipedia_WeightFactor_level2>0.5</
        IDFentireWikipedia_WeightFactor_level2>
    </wikiindex.characteristic.TVChar_IDFentireWikipedia>
   </TermVectorChars>
  </wikiindex.termvector.MultipleWeightSparseTermVectorType>
  <wikiindex.termvector.MultipleWeightSparseTermVectorType>
   <wikiindex.characteristic.TV_Char>TF</wikiindex.characteristic.TV_Char>
   <wikiindex.characteristic.TV_Scope>entity</wikiindex.characteristic.
```

```
        TV_Scope>
      <wikiindex.characteristic.TV_UseType>training</wikiindex.characteristic.
        TV_UseType>
      <LinkType>
       <wikiindex.characteristic.TV_LinkIN>
        <LinkIN_WeightFactor_level0>0.5</LinkIN_WeightFactor_level0>
        <LinkIN_WeightFactor_level1>0.5</LinkIN_WeightFactor_level1>
        <LinkIN_articleSelectionStrategy>firstn</
          LinkIN_articleSelectionStrategy>
        <LinkIN_crawlingDepth>1</LinkIN_crawlingDepth>
        <LinkIN_maxLinksToFollow>20</LinkIN_maxLinksToFollow>
        <LinkIN_weightingFactor>0.6</LinkIN_weightingFactor>
        <LinkIN_aggregationType>WeightedGeometricAverage</
          LinkIN_aggregationType>
       </wikiindex.characteristic.TV_LinkIN>
      </LinkType>
      <TermVectorChars>
       <wikiindex.characteristic.TVChar_TermFrequency>
        <TermFrequency_WeightFactor_level0>0.6</
          TermFrequency_WeightFactor_level0>
        <TermFrequency_WeightFactor_level1>0.5</
          TermFrequency_WeightFactor_level1>
       </wikiindex.characteristic.TVChar_TermFrequency>
       <wikiindex.characteristic.TVChar_IDFtrainingSet>
        <IDFtrainingSet_WeightFactor_level0>0.4</
          IDFtrainingSet_WeightFactor_level0>
        <IDFtrainingSet_WeightFactor_level1>0.5</
          IDFtrainingSet_WeightFactor_level1>
       </wikiindex.characteristic.TVChar_IDFtrainingSet>
      </TermVectorChars>
     </wikiindex.termvector.MultipleWeightSparseTermVectorType>
    </wikiindex.termvector.MultipleWeightSparseTermVectorType>
   </wikiindex.config.EntityClassifierTrainingConfig>
  </wikiindex.config.EntityClassifierTrainingConfig>
  <wikiindex.config.EntityClassifierTestingConfig>
   <wikiindex.config.EntityClassifierTestingConfig>
    <entityNamesAreWikipediaArticleTitles>true</
      entityNamesAreWikipediaArticleTitles>
    <skipUnresolvedTitles>true</skipUnresolvedTitles>
    <testingNBestToRetainInProtocol>10</testingNBestToRetainInProtocol>
    <wikiindex.termvector.MultipleWeightSparseTermVectorType>
     <wikiindex.termvector.MultipleWeightSparseTermVectorType>
      <wikiindex.characteristic.TV_Char>TF</wikiindex.characteristic.TV_Char>
      <wikiindex.characteristic.TV_Scope>entity</wikiindex.characteristic.
        TV_Scope>
      <wikiindex.characteristic.TV_UseType>testing</wikiindex.characteristic.
        TV_UseType>
```

```
    <LinkType>
     <wikiindex.characteristic.TV_LinkIN>
      <LinkIN_WeightFactor_level0>0.3</LinkIN_WeightFactor_level0>
      <LinkIN_WeightFactor_level1>0.5</LinkIN_WeightFactor_level1>
      <LinkIN_WeightFactor_level2>0.2</LinkIN_WeightFactor_level2>
      <LinkIN_articleSelectionStrategy>firstn</
          LinkIN_articleSelectionStrategy>
      <LinkIN_crawlingDepth>2</LinkIN_crawlingDepth>
      <LinkIN_maxLinksToFollow>20</LinkIN_maxLinksToFollow>
      <LinkIN_weightingFactor>1.0</LinkIN_weightingFactor>
      <LinkIN_aggregationType>WeightedGeometricAverage</
          LinkIN_aggregationType>
     </wikiindex.characteristic.TV_LinkIN>
    </LinkType>
    <TermVectorChars>
     <wikiindex.characteristic.TVChar_TermFrequency>
      <TermFrequency_WeightFactor_level0>1.0</
          TermFrequency_WeightFactor_level0>
      <TermFrequency_WeightFactor_level1>1.0</
          TermFrequency_WeightFactor_level1>
      <TermFrequency_WeightFactor_level2>1.0</
          TermFrequency_WeightFactor_level2>
     </wikiindex.characteristic.TVChar_TermFrequency>
    </TermVectorChars>
   </wikiindex.termvector.MultipleWeightSparseTermVectorType>
  </wikiindex.termvector.MultipleWeightSparseTermVectorType>
  <wikiindex.config.EntityClassifierSearchConfig>
   <wikiindex.config.EntityClassifierSearchConfig>
    <disambiguationCutoff>1</disambiguationCutoff>
    <pathToFileWithLuceneArticleKeys>searchkeys.csv</
        pathToFileWithLuceneArticleKeys>
    <searchProvider>KeysFromFile</searchProvider>
    <searchType>search</searchType>
   </wikiindex.config.EntityClassifierSearchConfig>
  </wikiindex.config.EntityClassifierSearchConfig>
 </wikiindex.config.EntityClassifierTestingConfig>
</wikiindex.config.EntityClassifierTestingConfig>
</evaluation.BOAExperimentConfig>
```

B.5. SCM Experiment Config Example

Below is a sample configuration for a WordSim353 experiment with SCM using all JWordnet-Sim measures (JWNL in the config file) and all JWSL measures using the most frequent sense strategy for both libraries.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<evaluation.WordSim353WordNetConfig>
 <additionalDebugFilesDir>debug-details</additionalDebugFilesDir>
 <consoleLoggingEnabled>false</consoleLoggingEnabled>
 <debugLogPath>debug.log</debugLogPath>
 <experimentClass>evaluation.WordSim353WordNet</experimentClass>
 <experimentName>testExperiment</experimentName>
 <groundtruthFile>combined.csv</groundtruthFile>
 <groundtruth_col>2</groundtruth_col>
 <loggerLevel>INFO</loggerLevel>
 <protocolPath>protocolf1.csv</protocolPath>
 <serializationPath>serializedClassifier</serializationPath>
 <test_col>0</test_col>
 <test_from>1</test_from>
 <test_to>353</test_to>
 <train_col>1</train_col>
 <train_from>1</train_from>
 <train_to>353</train_to>
 <trainingFile>combined.csv</trainingFile>
 <testingFile>combined.csv</testingFile>
 <wikiindex.config.SerializationPolicyEnum>noserialize</wikiindex.config.
    SerializationPolicyEnum>
 <useJWSL>true</useJWSL>
 <useJWNL>true</useJWNL>
 <similarityFunction>dotProduct</similarityFunction>
 <JWSL_wordnetLucenefolder>wn_index</JWSL_wordnetLucenefolder>
 <JWNL_WordnetICfolder></JWNL_WordnetICfolder>
 <JWNL_infoContentFileName>ic-bnc-resnik-add1.dat</JWNL_infoContentFileName>
 <JWNL_jwnlinitPath>map_propertiesWN20.xml</JWNL_jwnlinitPath>
 <JWSLMeasures>Resnik;Jiang;Lin;Pirro and Seco</JWSLMeasures>
 <JWNLMeasures>shef.nlp.wordnet.similarity.JCn;shef.nlp.wordnet.similarity.Lin
    </JWNLMeasures>
 <JWSL_senseSelectionStrategy>MostFrequentSense</JWSL_senseSelectionStrategy>
 <JWNL_senseSelectionStrategy>MostFrequentSense</JWNL_senseSelectionStrategy>
</evaluation.WordSim353WordNetConfig>
```

B.6. Other Configuration Files

The configuration XML file references four file paths `training.csv`, `testing.csv`, `groundtruth.csv` and `searchkeys.csv`.

There is also a column number associated with each of the first three files, which allows to use only one file and store the information in different columns. In all these files semicolon is used to separate columns. The use of the first three files depends on the activated `ExperimentConfig` class as denoted by the root element of the configuration file. We will therefore first describe the `searchkeys.csv`, which is common for all experiment types.

The `searchkeys.csv` file is used to map a noun phrase to an entity article. Each line corresponds to one mapping, the first entry is the noun phrase and the second entry the title of the entity article. This file is used for benchmarking to avoid repeated time-intensive disambiguation of the same noun phrase. The use of this file is setup independently for test and training phase, one file can also be used for both phases.

B.6.1. Word Similarity Computation Task

This section applies to `BOAExperimentConfig` and `SCMExperimentConfig` experiment types.

- `training.csv` lists target classes. The name of the class is extracted from column on position `train_entityname_col` which is optionally followed by the name of one entity article in `train_col`. If the second column is missing, the name of the class is interpreted as a noun phrase. Only the lines with numbers falling in range of the `train_from` and `train_to` parameters (zero-based, inclusive the bounds) are processed.
- `testing.csv` lists unlabeled instances (noun phrases) in col `test_col`. Only the lines with numbers falling in range of the `test_from` and `test_to` parameters (zero-based, inclusive the bounds) are processed.
- `groundtruth.csv` contains in column `groundtruth_col` the name of the correct target class for the unlabeled instance identified by the line number. Only the lines with numbers falling in range of the `test_from` and `test_to` parameters (zero-based, inclusive the bounds) are processed.

These files for the Toy example look as follows. For `groundtruth.csv` we assume that class 1 was given as correct class for the testing entity.

Listing B.1: training.csv

```
class 1;a1
class 2;a3
```

Listing B.2: testing.csv

```
t5 t6 t8
```

Listing B.3: groundtruth.csv

```
class 1
```

Listing B.4: searchkeys.csv

```
"t5 t6 t8";"a5"
"class 1";"a1"
```

B.6.2. Classification Task

This section applies to `WordSim353Config` and `WordSim353WordNetConfig` experiment types.

- `training.csv` lists the first word in the pair identified by the line number.
- `testing.csv` lists the second word in the pair identified by the line number.
- `groundtruth.csv` lists the average similarity value for the pair identified by the line number

Only the lines with numbers falling in range of the `test_from` and `test_to` parameters (zero-based, inclusive the bounds) are processed. The `train_from` and `train_to` parameters are ignored.

B.7. Creating the Index

The implementation offers a utility which creates a Lucene "Wikipedia" index from arbitrary data provided by the user. This index can be used in place of the index produced by Lucene-Search Mediawiki Extension from Wikipedia dumps.

The Index utility is executed by running `WikiIndex.jar` program with one parameter – path to the root directory with index files.

```
java -jar WikiIndex data/microtest/
```

The input data for the utility need to be placed in the `docs` and `linkdocs` subdirectories of the `root` directory. This structure is for the Toy example as follows:

```
root (dir)
 - wiki  (dir)
   - docs (dir)
     - a1.1.cat (file)
     - a2.2.cat (file)
     - a3.3.cat (file)
     - a4.4.cat (file)
     - a5.5.cat (file)
     - a6.6.cat (file)
   - linkdocs (dir)
     - a1.1 (file)
     - a2.2 (file)
     - a3.3 (file)
     - a4.4 (file)
     - a5.5 (file)
     - a6.6 (file)
```

The tool supports three modalities: in-link, out-link and same category.

The `docs` directory contains files with the following mask: `article name.id.category name*`. Each file (article) needs to have at least one category, however multiple categories are also allowed. The content of the files corresponds to the content of the articles. For example, the content of the file `a1.1.cat`:

```
t1 t2 t2
```

The files in the `linkdocs` directory follow the mask `articlename.id` and contain the links that lead from the article identified by the filename. The target articles are listed one per line and are identified by the article title.

For example, the content of the file `a1.1`:

```
a2
a3
```

The output of the program are `wiki` and `wiki.links` subdirectories created in the root directory containing the Main and Links Lucene indexes.

C. WordSim353 Dataset

Table C.1.: List of word pairs from the WordSim353 dataset – Set 1, part 1

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13
love	sex	6.77	9	6	8	8	7	8	8	4	7	2	6	7	8
tiger	cat	7.35	9	7	8	7	8	9	8.5	5	6	9	7	5	7
tiger	tiger	10	10	10	10	10	10	10	10	10	10	10	10	10	10
book	paper	7.46	8	8	7	7	8	9	7	6	7	8	9	4	9
computer	keyboard	7.62	8	7	9	9	8	8	7	7	6	8	10	3	9
computer	internet	7.58	8	6	9	8	8	8	7.5	7	7	7	9	5	9
plane	car	5.77	6	6	7	5	3	6	7	6	6	6	7	3	7
train	car	6.31	7	7.5	7.5	5	3	6	7	6	6	6	9	4	8
telephone	communication	7.5	7	6.5	8	8	6	8	8	7	5	9	9	8	8
television	radio	6.77	7	7.5	9	7	3	6	7	8	5.5	6	8	6	8
media	radio	7.42	7	7	8.5	9	6	7	7	7	5	7	10	8	8
drug	abuse	6.85	7	5.5	8	10	7	9	7	7	4	5	8.5	4	7
bread	butter	6.19	6	5.5	8	9	6	8	7	5	6	4	4	3	9
cucumber	potato	5.92	7	7.5	7	6	4	6	6.5	4	6	4	5	6	8
doctor	nurse	7	7	8	9	7	5	7	7	7	6	6	7	7	8
professor	doctor	6.62	6	7.5	8	7	4	6	8.5	6	7	3	6	9	8
student	professor	6.81	6	8	9	6	4	8	7.5	8	6	3	8	7	8
smart	student	4.62	5	6	5	2	4	6	5	7	4	3	3	5	5
smart	stupid	5.81	3	7	9	9	5	8	6	0	6.5	3	5	7	7
company	stock	7.08	6	8	9	9	4	8	8	6	6	8	6	6	8
stock	market	8.08	8	9	9.5	8	5	9	9	9	7	8	7.5	7	9
stock	phone	1.62	3	1	0	1	4	3	1	1	1	3	0	2	1
stock	CD	1.31	2	1	0	1	4	1	0	0	1	3	3	1	0
stock	jaguar	0.92	1	0	0	1	4	0	2	0	1	3	0	0	0
stock	egg	1.81	5	1.5	0	1	3	0	0	2	2	3	1	5	0
fertility	egg	6.69	7	8	8	7	4	8	8	8	6	9	2	6	6
stock	live	3.73	8	5.5	6	1	4	5	0	0	1	6	2	3	7
stock	life	0.92	4	0	0	1	3	0	0	0	2	0	0	0	2
book	library	7.46	8	9	8	9	5	9	8	7	6	8	7	6	7
bank	money	8.12	8	9	9.5	9	5	9	8	9	6	9	9	8	7
wood	forest	7.73	9	6	9.5	7	5	9	9	9	6	8	9	8	6
money	cash	9.15	10	9.5	9.5	10	8	9	9.5	10	7	10	8.5	9	9
professor	cucumber	0.31	1	0	0	1	2	0	0	0	0	0	0	0	0
king	cabbage	0.23	1	0	0	1	1	0	0	0	0	0	0	0	0
king	queen	8.58	9	9.5	9.5	10	7	8	8.5	9	8	8	8	8	9
king	rook	5.92	7	7.5	7	6	7	8	8.5	0	8	0	6	5	7
bishop	rabbi	6.69	7	7.5	8.5	2	5	7	9	7	8	7	6	7	6
Jerusalem	Israel	8.46	9	8	9.5	9	8	8	8	8	9	8	9.5	7	9
Jerusalem	Palestinian	7.65	9	8	9	8	4	8	8	7	9	8	9.5	5	7
holy	sex	1.62	3	0	5	2	1	1	7	0	1	0	0	0	1
fuck	sex	9.44	10	9.75	10	10	8	9	9	10	9	10	8	10	10
Maradona	football	8.62	9	9	9.5	10	7	9	8.5	8	9	10	9	6	8
football	soccer	9.03	9	9.9	9	10	8	7	9.5	10	8	10	9	9	9
football	basketball	6.81	8	7.5	8.5	3	3	8	8	6	7.5	8	7	6	8
football	tennis	6.63	7	7.25	8.5	3	3	7	8	6	7.5	8	7	6	8
tennis	racket	7.56	4	8	9.5	9	4	9	7.5	8	7.8	8	8.5	6	9
Arafat	peace	6.73	8	8	9.5	9	2	8	7	7	7	5	7	4	6
Arafat	terror	7.65	8	8	9.5	9	5	9	7	7	8	10	8	4	7
Arafat	Jackson	2.5	4	7.5	0	2	3	1	0	0	4	1	6	2	2
law	lawyer	8.38	9	8	9.5	9	6	9	9.5	9	7	10	6	8	9
movie	star	7.38	9	8	9.5	8	6	9	7	8	7	5	6.5	5	8
movie	popcorn	6.19	7	6	9	4	5	7	8	9	6	4	6.5	2	7
movie	critic	6.73	8	7.5	9.5	9	6	8	7.5	8	5	4	5	3	7
movie	theater	7.92	8	8	9.5	9	6	8	8.5	7	7	10	6	7	9
physics	proton	8.12	9	8.5	9	10	6	8	8.5	8	7	8	9.5	5	9
physics	chemistry	7.35	8	8.5	9	8	5	7	8	7	8	7	7	4	9
space	chemistry	4.88	6	8	3	5	5	6	7.5	6	5	3	1	2	6
alcohol	chemistry	5.54	8	8	7	5	5	8	6	4	4	2	6	5	4
vodka	gin	8.46	9	9.5	9	10	7	8	8.5	9	8	10	6	8	8
vodka	brandy	8.13	9	9.25	9	7	7	8	8.5	9	8	10	6	7	8
drink	car	3.04	7	0.5	0	2	2	7	5	5	4	0	2	0	5
drink	ear	1.31	3	0	0	5	2	2	2	0	1	1	0	0	1
drink	mouth	5.96	8	6	7	7	3	7	4	6	7.5	6	6	2	8
drink	eat	6.87	8	8.75	9	9	3	7	8	9	7.5	0	5	7	8
baby	mother	7.85	9	9	9	9	5	8	7	8	8	10	8	3	9
drink	mother	2.65	3	1	1	3	3	7	0	2	6.5	0	4	1	3
car	automobile	8.94	9	9.75	10	10	5	6	10	10	7.5	10	10	10	9
gem	jewel	8.96	9	9.5	10	10	6	8	8.5	10	7.5	10	10	8	10
journey	voyage	9.29	9	9.75	10	10	6	7	9.5	10	9.5	10	10	10	10
boy	lad	8.83	9	9.75	10	10	6	5	9.5	9	7.5	10	10	9	10
coast	shore	9.1	9	9.75	10	10	6	6	10	8	9.5	10	10	10	10
asylum	madhouse	8.87	9	9.75	7	10	6	7	9.5	10	9	10	10	9	9
magician	wizard	9.02	9	9.75	10	10	7	6	9	8	8.5	10	10	10	10
midday	noon	9.29	10	9.75	9	10	7	6	9.5	10	9.5	10	10	10	10
furnace	stove	8.79	9	9.75	9.5	10	7	7	9	9	8	10	10	8	8
food	fruit	7.52	8	8.75	8	8	5	7	8	7	7	9	8	7	7

Table C.2.: List of word pairs from the WordSim353 dataset – Set 1, part 2

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13
bird	cock	7.1	8	8.5	8	7	4	7	7	6	6.8	9	8	7	6
bird	crane	7.38	9	8.5	8.5	7	4	7	7	7	7	9	8	7	7
tool	implement	6.46	8	6	8	7	4	7	6	6	5	6	10	7	4
brother	monk	6.27	8	7	8	8	5	7	8.5	7	5	0	8	5	5
crane	implement	2.69	3	6	1	1	4	1	0	6	1	0	9	3	0
lad	brother	4.46	8	5.5	5	2	4	3	7	4	2.5	7	5	2	3
journey	car	5.85	7	7	7	6	4	6	6	7	5	5	5	4	7
monk	oracle	5	7	8	3	4	4	6	5	8	6	3	4	6	1
cemetery	woodland	2.08	3	2	1	2	3	6	2	3	3	0	0	1	1
food	rooster	4.42	7	4	4	6	3	6	7	2	4.5	2	8	1	3
coast	hill	4.38	6	6	6	5	2	6	5	5	4	3	4	1	4
forest	graveyard	1.85	4	2	1	1	2	6	1	3	3	0	0	1	0
shore	woodland	3.08	6	6	1	1	2	6	5	4	4	3	0	1	1
monk	slave	0.92	3	2	1	1	2	2	0	0	0	0	0	1	0
coast	forest	3.15	6	6	1	1	2	6	5	4	4	3	1	1	1
lad	wizard	0.92	4	0	0	1	2	3	0	0	0	0	0	1	1
chord	smile	0.54	3	0	0	1	2	1	0	0	0	0	0	0	0
glass	magician	2.08	4	1	4	2	2	3	0	0	0	0	10	0	1
noon	string	0.54	2	0	0	1	2	1	0	0	1	0	0	0	0
rooster	voyage	0.62	2	0	0	1	2	1	0	0	1	0	1	0	0
money	dollar	8.42	9	9.5	9	10	5	8	8.5	8	8	10	8.5	8	8
money	cash	9.08	10	9.5	9.5	10	5	9	9.5	10	8	10	8.5	9	10
money	currency	9.04	10	9.5	9	10	5	9	9.5	9	9	10	8.5	9	10
money	wealth	8.27	9	8	9	9	5	6	8.5	9	9	10	8	9	8
money	property	7.57	8	8.25	6	8	5	8	8	7	8.2	8	8	7	9
money	possession	7.29	8	8.25	7	7	5	7	7.5	5	9	7	8	7	9
money	bank	8.5	9	8	9.5	9	6	9	8.5	9	8.5	10	8	7	9
money	deposit	7.73	9	8	9.5	9	5	9	8	7	7	8	7	7	7
money	withdrawal	6.88	8	8	9	9	5	9	8	5	6.5	8	2	5	7
money	laundering	5.65	8	7	8	7	5	7	7.5	2	5	5	5	0	7
money	operation	3.31	4	2	3	5	5	5	2	2	4	3	5	1	2
tiger	jaguar	8	9	9	9	10	5	8	7.5	6	8	9	8.5	7	8
tiger	feline	8	9	7.5	9.5	8	5	8	8.5	8	8	9	8.5	7	8
tiger	carnivore	7.08	9	6	8	5	5	8	7	7	8.1	8	6	7	8
tiger	mammal	6.85	9	7	7.5	5	5	7	7	7	8	8	8.5	6	4
tiger	animal	7	8	7	7.5	5	5	6	6	7	7.5	9	10	5	8
tiger	organism	4.77	8	7	6	1	5	5	1	2	6	4	10	5	2
tiger	fauna	5.62	8	7	7.5	1	2	6	7	2	5.5	9	10	5	3
tiger	zoo	5.87	8	5.25	8	3	5	7	6	8	6	5	5	5	5
psychology	psychiatry	8.08	9	8.5	9.5	9	4	8	8	8	7	9	9	8	8
psychology	anxiety	7	7	7	8.5	4	5	8	5	7	8	8	8.5	7	8
psychology	fear	6.85	8	7	8.5	4	5	8	5	5	8	8	8.5	7	7
psychology	depression	7.42	9	8	9	5	5	8	5	7	8	8	8.5	7	9
psychology	clinic	6.58	8	8	9.5	5	4	9	7	6	6	7	6	4	6
psychology	doctor	6.42	9	8	9	5	5	8	7.5	6	5	5	7	4	5
psychology	Freud	8.21	9	9.25	9.5	10	5	8	8	9	7.5	9	8.5	7	7
psychology	mind	7.69	9	9.5	9	9	5	8	5	8	7	8	8.5	7	7
psychology	health	7.23	9	8	8.5	5	5	7	5	8	8	8	8.5	6	8
psychology	science	6.71	8	7.75	7.5	5	5	7	4	8	6.5	7	9.5	4	8
psychology	discipline	5.58	8	7	7.5	6	4	6	4	7	6	8	2	5	2
psychology	cognition	7.48	9	9.75	8.5	6	3	8	8	8	6.5	9	7.5	8	6
planet	star	8.45	9	9.9	10	10	5	8	8.5	9	8	9	8.5	7	8
planet	constellation	8.06	9	8.25	9	9	6	8	8	8	8	9	8.5	6	8
planet	moon	8.08	9	8.5	9	9	5	8	7	9	8	10	8.5	7	7
planet	sun	8.02	9	8.75	9	8	5	8	7	9	8	10	8.5	7	7
planet	galaxy	8.11	9	8.25	9.5	8	5	8	8	8	8.2	10	9.5	6	8
planet	space	7.92	9	8	9.5	6	5	8	7	8	8	10	9.5	6	9
planet	astronomer	7.94	9	8.25	9.5	8	5	8	7	7	8	10	7.5	7	9
precedent	example	5.85	6	8.5	9.5	10	3	7	6	8	1	1	9	2	5
precedent	information	3.85	5	5.5	6	8	4	5	1	2	0	1	7.5	2	3
precedent	cognition	2.81	6	5.5	2	7	3	3	0	0	0	1	7	1	1
precedent	law	6.65	8	7	8.5	8	3	8	7	5	7.5	8	8.5	5	3
precedent	collection	2.5	7	5.5	1	3	3	5	0	0	0	0	6	1	1
precedent	group	1.77	6	1	1	2	3	5	0	0	0	0	4	1	0
precedent	antecedent	6.04	9.5	0	9.5	9	4	7	7.5	7	8	7	5	3	2
cup	coffee	6.58	9	8	9	8	5	9	8	5	6.5	5	4	3	6
cup	tableware	6.85	7	8.5	8	9	5	7	7	4	5	9	8.5	5	6
cup	article	2.4	1	8.25	6	3	3	0	0	0	0	3	3	2	2
cup	artifact	2.92	7	8	1	2	4	0	2	1	5	3	4	0	1
cup	object	3.69	8	8	4	2	5	3	2	5	5	0	3	2	1
cup	entity	2.15	3	6	3	1	4	3	0	3	4	0	1	0	0
cup	drink	7.25	9	7.75	9	8	4	8	6	8	8	6	7.5	6	7
cup	food	5	7	7	6	2	3	7	3	7	6	4	4	4	5
cup	substance	1.92	3	3	2	1	3	2	1	3	5	0	2	0	0
cup	liquid	5.9	9	7.75	7	5	4	7	4	7	7	6	7	4	2
jaguar	cat	7.42	9	7	8	8	4	8	7.5	7	7	9	7	7	8
jaguar	car	7.27	9	9	8.5	8	4	8	7	7	8	9	4	5	8

Table C.3.: List of word pairs from the WordSim353 dataset – Set 2, part 1

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
energy	secretary	1.81	1	0	4	2	4	5	1	1	1	0	1	1	4	0	2	2
secretary	senate	5.06	7	1	7	4	4	7	1	3	4	8	4	5	6	7	6	7
energy	laboratory	5.09	7	1	7.5	4	6	7	4	6	1	2	4	3	7	9	6	7
computer	laboratory	6.78	8	5	8	7	6	9	6	7	6	7.5	4	5	8	9	7	6
weapon	secret	6.06	7	4	8	6	6	9	2	6	3	6	5	6	8	9	7	5
FBI	fingerprint	6.94	8	6	8	5	5	9	7	7	6	6	6	8	9	7	6	8
FBI	investigation	8.31	9	9	8.5	9	7	9	8	8	8	7.5	6	9	10	9	7	9
investigation	effort	4.59	5	1	7.5	2	4	7	6	5	2	2	2	7	6	5	6	6
Mars	water	2.94	2	1	3	2	1	8	0	4	2	6	1	1	3	0	5	8
Mars	scientist	5.63	8	1	7	4	6	8	1	6	5	6	2	9	7	5	7	8
news	report	8.16	9	6	8.5	8	7	9	7	8	7	8	7	9	10	9	9	9
canyon	landscape	7.53	9	7.5	7	7	7	7	8	7	7	8	6	9	8	6	8	9
image	surface	4.56	7	1	5	1	1	5	4	3	4	4	5	5	7	7	8	6
discovery	space	6.34	8	2	7.5	9	5	7	4	7	5	6	5	7	8	8	7	6
water	seepage	6.56	8	7	9	7	7	8	7	6	0	6	1	8	7	8	8	8
sign	recess	2.38	4	1	2	4	5	4	0	4	3	0	1	0	0	0	6	4
Wednesday	news	2.22	4	1	4	2	3	6	2	3	0.5	0	1	1	0	1	4	3
mile	kilometer	8.66	9	9.5	9	8	9	8	9	9	8.5	7.5	8	8	10	8	9	9
computer	news	4.47	5	1	7	6	5	5	1	4	6.5	4	2	5	3	7	6	4
territory	surface	5.34	6	2	8.5	4	7	7	8	4	2	6	6	5	6	2	8	4
atmosphere	landscape	3.69	7	0	2	1	8	7	1	5	2	0	2	5	6	1	7	5
president	medal	3	5	2	1	3	6	6	1	7	2	0	1	3	2	1	4	4
war	troops	8.13	8	8.5	9	9	8	8	8	8	6	7.5	8	8	9	8	9	8
record	number	6.31	8	6	8	5	7	7	3	4	5	8	5	8	8	5	8	6
skin	eye	6.22	7	9	7	3	6	6	7	7	6	7.5	5	6	8	2	8	5
Japanese	American	6.5	7	6	8.5	6	6	4	8	7	7	7.5	5	8	9	1	8	6
theater	history	3.91	5	6	6	4	5	3	0	3	6.5	6	1	5	1	1	7	3
volunteer	motto	2.56	2	5	1	1	4	2	0	3	3	0	4	7	0	3	4	2
prejudice	recognition	3	7	4	2	1	6	2	4	5	1	0	6	5	0	0	3	2
decoration	valor	5.63	6	8	7	8	8	2	2	9	5	0	7	8	9	1	8	2
century	year	7.59	8	9	8	9	7	6	8	8	7	7.5	7	9	8	5	9	6
century	nation	3.16	5	0	7.5	5	4	4	0	2	2	4	2	2	2	0	8	3
delay	racism	1.19	1	0	1	1	6	1	0	1	0	0	4	0	0	0	3	1
delay	news	3.31	7	4	3	1	6	5	6	4	4	0	1	4	3	0	3	2
minister	party	6.63	8	7.5	8	7	6	7	8	8	5	7.5	6	1	8	8	8	3
peace	plan	4.75	5	2	7	6	7	4	3	5	4	4	5	3	7	2	9	3
minority	peace	3.69	6	0	7	3	1	5	6	4	4	4	3	3	5	0	5	3
attempt	peace	4.25	7	4	7	2	7	4	2	3	3	4	3	5	5	4	5	3
government	crisis	6.56	8	5	8	5	7	7	5	6	5	6	7	8	7	9	7	5
deployment	departure	4.25	7	0	2	6	7	5	6	4	2	2	5	6	2	0	8	6
deployment	withdrawal	5.88	9	9	6	6	3	8	8	4	3	2	5	8	9	0	8	6
energy	crisis	5.94	8	5	8	5	8	8	8	4	1	4	2	7	5	8	8	6
announcement	news	7.56	8	7	8	8	8	9	8	6	6	8	8	8	6	8	9	6
announcement	effort	2.75	5	6	2	2	1	5	4	2	2	2	2	0	3	0	5	3
stroke	hospital	7.03	9	8	8	8	7	3	8	7	6	7.5	7	7	9	3	8	7
disability	death	5.47	7	8	6.5	4	3	7	7	5	6	2	7	2	8	2	8	5
victim	emergency	6.47	8	7	7.5	4	5	6	6	7	6	4	5	6	9	9	9	5
treatment	recovery	7.91	8	8	8.5	9	9	8	9	8	6.5	7.5	7	7	10	5	9	7
journal	association	4.97	8	2	7.5	7	1	5	4	6	3	6	7	3	7	1	8	4
doctor	personnel	5	7	2	8	6	4	7	2	6	2	6	4	5	7	3	8	3
doctor	liability	5.19	7	4	8	7	4	6	4	5	4	2	4	2	6	8	7	5
liability	insurance	7.03	6	8.5	9	5	8	9	6	8	5	4	7	8	8	5	9	7
school	center	3.44	6	1	4	1	5	2	4	3	3	4	5	1	3	3	7	3
reason	hypertension	2.31	4	1	1	2	6	1	0	1	3	2	2	7	0	2	3	2
reason	criterion	5.91	3	2	8	7	9	3	6	4	4.5	8	7	7	8	6	7	5
hundred	percent	7.38	9	5	9	3	10	9	6	7	7	6	6	7	10	9	8	7
Harvard	Yale	8.13	9	8	9	10	8	8	9	10	8.5	7.5	8	8	10	0	8	9
hospital	infrastructure	4.63	5	2	1	5	6	5	4	7	3	4	5	5	5	3	7	7
death	row	5.25	2	7	8	2	8	7	6	3	7	4	2	7	9	0	8	4
death	inmate	5.03	4	5	7.5	1	5	4	7	3	6	2	6	6	3	8	8	5
lawyer	evidence	6.69	7	6.5	8.5	7	8	6	8	8	7	3	5	8	10	0	9	6
life	death	7.88	9	9.5	9.5	5	10	8	8	8	6.5	7.5	8	9	10	0	9	9
life	term	4.5	2	8	2	1	8	7	6	3	1	4	2	8	3	3	8	6
word	similarity	4.75	6	1	8	2	10	7	6	2	1	4	2	5	0	9	7	6
board	recommendation	4.47	6	4	2	1	7	8	3	4	1	7.5	2	7	7	0	7	5

Table C.4.: The list of word pairs from the WordSim353 dataset – Set 2, part 2

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
governor	interview	3.25	4	0	4	1	7	6	0	5	0	2	3	5	3	3	4	5
OPEC	country	5.63	7	4	8	4	0	4	8	6	5	8	4	6	7	8	6	5
peace	atmosphere	3.69	6	5	5	1	0	6	3	5	2	6	1	3	3	0	7	6
peace	insurance	2.94	6	4	1	1	7	4	0	3	2	6	5	1	0	0	3	4
territory	kilometer	5.28	6	6	8	1	5	8	7	6	4	7.5	5	3	3	2	8	5
travel	activity	5	7	5	6.5	2	5	6	6	6	5	7.5	3	2	4	2	8	5
competition	price	6.44	7	8	7.5	5	6	7	8	4	6	7.5	5	6	9	5	9	3
consumer	confidence	4.13	7	4	2	3	3	3	1	2	2	6	2	7	9	4	8	3
consumer	energy	4.75	5	3	8	2	6	6	0	4	0	4	7	6	6	9	8	2
problem	airport	2.38	2	2	1	0	5	4	0	3	0	4	1	2	0	7	5	2
car	flight	4.94	6	3	7	5	6	6	4	7	4	2	7	5	8	0	5	4
credit	card	8.06	8	6	9	8	9	8	8	9	7	8	5	8	10	9	9	8
credit	information	5.31	7	5	2	3	7	4	1	5	7	1	2	8	10	9	8	6
hotel	reservation	8.03	8	7	9	7	8	8	8	8	7	7.5	7	8	10	9	9	8
grocery	money	5.94	8	5	7.5	2	7	7	7	5	6.5	6	6	8	6	2	5	7
registration	arrangement	6	8	8	7	7	8	4	1	4	7	6	6	6	8	3	8	5
arrangement	accommodation	5.41	5	6	7	6	4	4	4	6	3	7.5	5	5	7	6	7	4
month	hotel	1.81	4	0	1	1	3	6	0	2	1	1	2	0	3	0	3	2
type	kind	8.97	9	9.5	9.5	10	8	9	9	10	8.5	9	9	9	9	9	9	7
arrival	hotel	6	7	8	6.5	1	6	9	7	5	6	7.5	4	6	7	4	6	6
bed	closet	6.72	8	7.5	8	5	7	8	7	8	7	6	7	5	9	1	8	6
closet	clothes	8	8	7	9	10	8	9	8	9	9	8	7	8	9	3	9	7
situation	conclusion	4.81	8	4	6	4	6	3	7	6	2	6	6	7	4	0	6	2
situation	isolation	3.88	8	3	1	2	6	3	0	3	2	6	6	7	3	0	8	4
impartiality	interest	5.16	7	9.5	8	2	4	3	0	7	6	3	7	6	9	0	8	3
direction	combination	2.25	2	0	1	1	7	2	0	2	2	2	4	3	0	1	6	3
street	place	6.44	7	7	6	7	5	7	5	7	6	4	7	8	4	7	9	7
street	avenue	8.88	9	9	8.5	9	9	9	7	10	8.5	9	7	9	10	9	10	9
street	block	6.88	5	7	8.5	7	4	8	7	9	8.5	9	6	9	4	0	9	9
street	children	4.94	6	6	6	2	5	7	2	5	4	6	3	5	6	5	5	6
listing	proximity	2.56	3	0	1	1	5	4	0	3	1	2	2	4	3	1	7	4
listing	category	6.38	2	3	7	9	7	6	7	7	4	8	7	8	10	3	8	6
cell	phone	7.81	8	6	8	9	9	8	8	8	8	8	3	9	8	9	8	8
production	hike	1.75	2	2	1	1	6	3	0	1	1	0	2	3	0	0	4	2
benchmark	index	4.25	5	5	2	2	7	4	5	7	3	1	7	2	6	4	5	3
media	trading	3.88	6	2	5	1	7	6	3	4	2	2	6	2	5	0	8	3
media	gain	2.88	5	0	2	1	7	4	2	2	3	2	5	1	3	0	7	2
dividend	payment	7.63	6	9	7	4	9	8	8	8	7	8	7	8	8	9	9	7
dividend	calculation	6.48	7	8.75	6.5	1	9	7	6	7	6.5	8	3	6	7	8	8	5
calculation	computation	8.44	9	9.5	9.5	8	8	9	5	10	9	8	8	9	7	9	9	8
currency	market	7.5	8	5	7.5	5	9	8	7	8	7	7.5	6	8	8	9	9	8
OPEC	oil	8.59	8	8	9	10	10	8	8	8	8	7.5	7	9	10	9	9	9
oil	stock	6.34	6	5	8	6	7	6	2	6	8	7.5	6	6	7	8	7	6
announcement	production	3.38	5	0	3	2	6	5	1	4	2	2	3	6	3	5	6	1
announcement	warning	6	7	7	5	7	8	4	4	8	5	7	6	8	3	8	7	2
profit	warning	3.88	7	7	1	5	4	6	4	3	5	1	3	3	0	0	7	6
profit	loss	7.63	8	9.5	9.5	10	8	8	8	9	6	4	8	8	10	0	9	7
dollar	yen	7.78	8	9	9	10	7	5	8	8	7	7.5	8	8	10	3	9	8
dollar	buck	9.22	9	10	9.5	10	9	8	9	10	9	8	10	10	8	9	10	9
dollar	profit	7.38	8	6	9	9	9	7	7	8	8	6	7	5	6	7	8	8
dollar	loss	6.09	8	6	8.5	2	5	7	7	8	4	6	5	5	5	7	8	6
computer	software	8.5	9	8	9	8	9	8	8	9	8	9	8	8	10	9	9	7
network	hardware	8.31	8	8.5	8.5	9	9	7	8	7	8	9	8	9	8	9	9	8
phone	equipment	7.13	8	8.5	7.5	5	9	6	7	6	6	8	5	7	7	9	8	7
equipment	maker	5.91	8	7	7	6	6	5	8	5	2	7.5	3	4	6	5	8	7
luxury	car	6.47	7	6	8	6	7	7	5	5	3	7.5	6	6	7	7	8	8
five	month	3.38	5	6	2	1	5	4	1	5	2	4	2	1	3	6	5	2
report	gain	3.63	6	5	4	1	4	5	4	3	2	0	2	1	7	7	6	1
investor	earning	7.13	8	7	8	8	7	7	8	7	3	8	7	6	8	7	8	7
liquid	water	7.89	8	7.75	8.5	8	7	8	4	8	7	9	8	9	8	9	9	8
baseball	season	5.97	8	3	8.5	7	8	8	4	6	5	8	3	6	7	2	7	5
game	victory	7.03	8	8	7	7	5	8	6	7	6.5	8	7	5	9	8	7	6
game	team	7.69	8	8.5	8.5	8	5	8	6	9	7	9	7	7	9	8	8	7
marathon	sprint	7.47	7	9	7	8	6	8	7	8	6.5	9	8	8	5	9	8	6
game	series	6.19	7	8	7.5	6	5	8	5	6	2	7.5	3	8	6	5	8	7
game	defeat	6.97	8	8	7.5	7	6	8	6	7	6	8	7	5	9	6	7	6
seven	series	3.56	5	6	2	2	5	4	4	5	3	4	4	1	0	1	7	4
seafood	sea	7.47	9	8	7.5	8	9	9	7	7	7	9	6	6	5	7	8	7
seafood	food	8.34	9	9.5	9	9	9	8	4	9	7	9	8	9	8	9	9	8

Table C.5.: The list of word pairs from the WordSim353 dataset – Set 2, part 3

Word 1	Word 2	Mean	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
seafood	lobster	8.7	9	9.75	8.5	10	9	9	6	10	7	9	9	9	8	9	9	8
lobster	food	7.81	8	8	8	10	9	7	4	9	7	9	6	9	9	7	8	7
lobster	wine	5.7	7	7.75	7.5	1	6	6	3	8	5.5	7.5	4	6	7	1	7	7
food	preparation	6.22	6	5	6.5	5	9	6	3	7	6	8	5	6	6	7	8	6
video	archive	6.34	7	6	7	5	9	7	4	7	4	7.5	5	7	5	9	8	4
start	year	4.06	5	5.5	6.5	2	5	3	0	6	4	6	2	2	3	9	4	2
start	match	4.47	5	2	6.5	1	9	3	3	8	1	2	3	5	5	9	6	3
game	round	5.97	4	8	8	4	5	5	7	9	4	7.5	5	6	7	5	8	3
boxing	round	7.61	6	8.25	7.5	7	8	7	8	9	4	8	7	8	10	9	9	6
championship	tournament	8.36	9	8.75	9	9	9	8	7	10	7	8	8	8	10	9	8	6
fighting	defeating	7.41	8	8	8	8	5	7	6	9	7	7.5	8	9	8	5	9	6
line	insurance	2.69	5	2	2	1	6	3	3	4	1	2	2	2	0	0	9	1
day	summer	3.94	7	7	2	1	4	3	1	7	3	4	4	4	6	1	5	4
summer	drought	7.16	8	8	8	9	5	7	7	8	5	7.5	5	4	8	9	8	8
summer	nature	5.63	5	6	4	7	9	5	6	7	4	6	6	3	3	6	7	6
day	dawn	7.53	8	8.5	8	8	9	8	6	8	6	6	8	7	8	7	8	7
nature	environment	8.31	9	6.5	8.5	9	9	9	7	9	8	8	8	8	10	9	8	7
environment	ecology	8.81	9	8.5	8.5	9	9	9	8	10	8	9	9	9	10	7	9	9
nature	man	6.25	5	6	8	5	9	5	5	7	3	8	5	6	8	5	8	7
man	woman	8.3	8	9.75	9	10	9	9	8	9	7.5	7.5	9	9	10	1	9	8
man	governor	5.25	7	4.5	5	1	6	4	3	7	2	7.5	3	7	7	8	7	5
murder	manslaughter	8.53	8	9.9	9.5	10	7	9	9	10	6	4	8	9	9	9	10	9
soap	opera	7.94	8	4	8.5	7	9	9	8	9	8	7.5	7	8	9	9	7	9
opera	performance	6.88	8	2	7.5	5	9	5	7	7	8	7.5	6	9	7	9	7	6
life	lesson	5.94	7	5	7.5	3	9	8	4	6	5	7.5	4	5	2	9	7	6
focus	life	4.06	5	0	7.5	1	7	6	0	3	4	7.5	3	2	2	7	5	5
production	crew	6.25	8	4	8	4	8	7	5	7	3	8	4	7	9	9	6	3
television	film	7.72	8	5	8.5	8	9	8	8	9	5	8	7	8	9	9	8	6
lover	quarrel	6.19	7	2	8.5	3	7	7	6	9	5	7.5	4	6	8	7	7	5
viewer	serial	2.97	3	0.5	1	1	4	3	0	6	2	2	2	7	7	3	4	2
possibility	girl	1.94	3	0	0	1	4	2	0	2	2	4	3	1	0	3	5	1
population	development	3.75	5	1	3	2	7	5	2	7	2	6	4	1	4	1	5	5
morality	importance	3.31	7	0	1	1	6	4	0	6	1	6	3	3	3	4	4	4
morality	marriage	3.69	6	5	2	1	5	4	1	3	3	6	3	5	4	1	4	6
Mexico	Brazil	7.44	8	8.5	8	6	7	7	8	9	7	7.5	7	8	10	2	8	8
gender	equality	6.41	8	6	5	4	4	8	5	8	6	7.5	2	8	8	7	7	9
change	attitude	5.44	8	5	6	3	4	7	4	7	7	6	4	6	5	0	7	8
family	planning	6.25	7	5	7	3	7	8	4	6	7	8	7	6	8	1	8	8
opera	industry	2.63	7	0	1	1	7	2	3	3	2	2	1	2	3	1	4	3
sugar	approach	0.88	3	0	0	1	5	1	0	1	0	0	1	0	0	0	1	1
practice	institution	3.19	4	1	1	1	6	2	2	2	4	3	6	6	5	0	5	3
ministry	culture	4.69	4	1	6	4	7	6	1	4	5	6	5	7	6	7	3	3
problem	challenge	6.75	7	7.5	7.5	8	7	7	1	10	5	6	6	8	8	8	7	5
size	prominence	5.31	7	8.5	6	7	6	4	2	8	3.5	2	6	5	6	1	7	6
country	citizen	7.31	8	7	7	9	8	7	7	7	5	8	7	8	9	6	8	6
planet	people	5.75	5	0	6	7	8	5	6	5	5	9	7	6	4	7	7	5
development	issue	3.97	5	1	6	1	5	3	1	5	3	7.5	3	7	3	2	5	6
experience	music	3.47	6	1	7	1	5	3	1	1	1	7.5	3	8	3	1	5	2
music	project	3.63	5	0	6	1	5	3	4	2	3	6	5	2	7	2	4	3
glass	metal	5.56	7	7	8	4	6	5	4	7	6	4	7	7	6	0	5	6
aluminum	metal	7.83	9	8.75	9	9	7	7	6	8	6	7.5	8	9	8	8	8	7
chance	credibility	3.88	8	3	1	3	6	3	3	4	3	6	3	8	1	1	4	5
exhibit	memorabilia	5.31	7	6	9	7	7	3	2	7	3	6	7	7	5	2	1	6
concert	virtuoso	6.81	8	6	7.5	9	6	7	7	8	2	7.5	7	7	8	3	8	8
rock	jazz	7.59	8	9	9	5	8	7	8	9	7	7.5	8	8	9	1	9	9
museum	theater	7.19	8	7.5	8.5	6	8	8	8	9	7	6	8	8	7	1	8	7
observation	architecture	4.38	7	3	3	4	6	3	6	5	3	4	7	7	3	1	2	6
space	world	6.53	7	6	8.5	3	9	6	7	9	5	6	8	8	5	3	8	6
preservation	world	6.19	7	8	7	6	7	7	2	8	4	6	6	8	8	4	6	5
admission	ticket	7.69	8	8	9	9	9	8	7	9	6	6	6	9	9	5	8	7
shower	thunderstorm	6.31	8	9	8	6	7	8	2	7	6	4	5	9	8	2	8	4
shower	flood	6.03	8	7	8.5	6	7	8	2	7	6	6	7	8	8	0	4	4
weather	forecast	8.34	8	7.5	9	9	9	9	7	9	7	8	8	9	10	7	9	8
disaster	area	6.25	7	8	9	3	9	8	3	5	6	6	5	8	9	1	5	8
governor	office	6.34	8	5	7.5	3	7	8	1	6	5	6	6	8	7	8	8	8
architecture	century	3.78	6	0	7.5	1	6	8	1	3	1	6	5	7	3	2	2	2

Table C.6.: WordSim353 entries mapped on Wikipedia articles – Part 1

Entry	Article title	Entry	Article title	Entry	Article title
Arafat	Arafat	equipment	Tool	population	Population
FBI	Federal Bureau of Investigation	exhibit	Exhibit	possibility	Possibility
Harvard	Harvard University	experience	Experience	practice	Practice
Japanese	Japanese	family	Family	precedent	Precedent
Jerusalem	Jerusalem	fertility	Fertility	prejudice	Prejudice
Maradona	Diego Maradona	fighting	Combat	preservation	Preservation
Mars	Mars	five	5 (number)	president	President
Mexico	Mexico	focus	Focus	problem	Problem
OPEC	OPEC	food	Food	production	Production
Wednesday	Wednesday	football	Football	professor	Professor
admission	Admission	forest	Forest	profit	Profit
alcohol	Alcohol	fuck	Fuck	psychology	Psychology
aluminum	Aluminium	furnace	Furnace	reason	Reason
announcement	Announcement (computing)	game	Game	record	Record
architecture	Architecture	gem	Gem	registration	Registration
arrangement	Arrangement	gender	Gender	report	Report
arrival	Arrival	glass	Glass	rock	Rock
asylum	Asylum	government	Government	rooster	Rooster
atmosphere	Atmosphere	governor	Governor	school	School
attempt	Attempt crime	grocery	Grocery store	seafood	Seafood
baby	Infant	holy	Sacred	secretary	Secretary
bank	Bank	hospital	Hospital	seven	7 (number)
baseball	Baseball	hotel	Hotel	shore	Shore
bed	Bed	hundred	Hundred	shower	Shower
benchmark	Benchmark	image	Image	sign	Dollar sign
bird	Bird	impartiality	Impartiality	situation	Situation
bishop	Bishop	investigation	Investigation	size	Size
board	Board	investor	Investor	skin	Skin
book	Book	jaguar	Jaguar	smart	S.M.A.R.T.
boxing	Boxing	journal	Journal	soap	Soap
boy	Boy	journey	Journey	space	Space
bread	Bread	king	King	start	Start
brother	Sibling	lad	LAD	stock	Stock
calculation	Calculation	law	Law	street	Street
canyon	Canyon	lawyer	Lawyer	stroke	Stroke
car	Automobile	liability	Liability	student	Student
cell	Cell (microprocessor)	life	Life	sugar	Sugar
cemetery	Cemetery	line	Line	summer	Summer
century	Century	liquid	Liquid	telephone	Telephone
championship	Championship	listing	Listing	television	Television
chance	Chance	lobster	Lobster	tennis	Tennis
change	Change	love	Love	territory	Territory
chord	Chord	lover	Intimate relationship	theater	Theatre
closet	Closet	luxury	Luxury	tiger	Tiger
coast	Coast	magician	Magician	tool	Tool
company	Company	man	Man	train	Train
competition	Competition	marathon	Marathon	travel	Travel
computer	Computer	media	Filesystem Hierarchy Standard	treatment	Treatment
concert	Concert	midday	Noon	type	Type
consumer	Consumer	mile	Mile	victim	Victim
country	Country	minister	Minister	video	Video
crane	Crane	ministry	Ministry	viewer	Viewer
credit	Credit	minority	Minority	vodka	Vodka
cucumber	Cucumber	money	Money	volunteer	Volunteering
cup	CUPS	monk	Monk	war	War
currency	Currency	month	Month	water	Water
day	Day	morality	Morality	weapon	Weapon
death	Death	movie	Film	weather	Weather
decoration	Decoration	murder	Murder	wood	Wood
delay	Tom DeLay	museum	Museum	word	Word
deployment	Deployment	music	Music	American	United States
development	Development	nature	Nature	Brazil	Brazil
direction	Direction	network	Network	CD	Compact Disc
disability	Disability	news	News	Freud	Sigmund Freud
disaster	Disaster	noon	Noon	Israel	Israel
discovery	Discovery	observation	Observation	Jackson	Jackson
dividend	Dividend	oil	Oil	Palestinian	Palestinian
doctor	Doctor	opera	Opera	Yale	Yale University
dollar	Dollar	peace	Peace	abuse	Abuse
drink	Drink	phone	Telephone	accommodation	Accommodation
drug	Drug	physics	Physics	activity	Activity
energy	Energy	plane	Plane	airport	Airport
environment	Environment	planet	Planet	animal	Animal

Table C.7.: WordSim353 entries mapped on Wikipedia articles – Part 2

Entry	Article title	Entry	Article title	Entry	Article title
antecedent	Antecedent	forecast	Forecasting	popcorn	Popcorn
anxiety	Anxiety	fruit	Fruit	possession	Possession
approach	Approach	gain	Gain	potato	Potato
archive	Archive	galaxy	Galaxy	preparation	Preparation
area	Area	gin	Gin	price	Price
article	Article	girl	Girl	project	Project
artifact	Artifact	graveyard	Graveyard	prominence	Topographic prominence
association	Association	group	Group (mathematics)	property	Property
astronomer	Astronomer	hardware	Hardware	proton	Proton
attitude	Attitude	health	Health	proximity	Distance
automobile	Automobile	hike	Hike	psychiatry	Psychiatry
avenue	Avenue	hill	Hill	quarrel	Quarrel
basketball	Basketball	history	History	queen	Queen (band)
block	Block	hypertension	Hypertension	rabbi	Rabbi
brandy	Brandy	implement	Implement	racism	Racism
buck	Buck	importance	Importance	racket	Racket
butter	Butter	index	Index	radio	Radio
cabbage	Cabbage	industry	Industry	recess	Recess
card	Card	information	Information	recognition	Recall (memory)
carnivore	Carnivore	infrastructure	Infrastructure	recommendation	Recommendation
cash	Cash	inmate	Prisoner	recovery	Recovery
cat	Cat	institution	Institution	reservation	Reservation
category	Category	insurance	Insurance	rook	Rook
center	Center	interest	Interest	round	Round
challenge	Challenge	internet	Internet	row	Row
chemistry	Chemistry	interview	Interview	science	Science
children	Child	isolation	Isolation	scientist	Scientist
citizen	Citizenship	issue	Issue	sea	Sea
clinic	Clinic	jazz	Jazz	season	Season
clothes	Clothing	jewel	Jewel	secret	Secrecy
cock	Cock	keyboard	Keyboard	seepage	Soil mechanics
coffee	Coffee	kilometer	Kilometre	senate	Senate
cognition	Cognition	kind	Kind	serial	Serial
collection	Collection	laboratory	Laboratory	series	Series
combination	Combination	landscape	Landscape	sex	Sex
communication	Communication	laundering	Laundering	similarity	Similarity
computation	Computation	lesson	Lesson	slave	Slavery
conclusion	Conclusion	library	Library	smile	Smile
confidence	Confidence	live	Live	soccer	Association football
constellation	Constellation	loss	Loss	software	Computer software
credibility	Credibility	madhouse	Madhouse	sprint	Sprint Nextel
crew	Crew	maker	Maker's Mark	star	Star
crisis	Crisis	mammal	Mammal	stove	Stove
criterion	Criterion	manslaughter	Manslaughter	string	String
critic	Critic	market	Market	stupid	Stupid!
culture	Culture	marriage	Marriage	substance	Substance
dawn	Dawn	match	Match	sun	Sun
defeat	Defeat	medal	Medal	surface	Surface
defeating	2003 NCAA Men's Division I Basketball Tournament	memorabilia	Souvenir	tableware	Tableware
departure	Departure	metal	Metal	team	Team
deposit	Deposit	mind	Mind	term	Term
depression	Depression	moon	Moon	terror	Terror
discipline	Discipline	mother	Mother	thunderstorm	Thunderstorm
drought	Drought	motto	Motto	ticket	Ticket
ear	Ear	mouth	Mouth	tournament	Tournament
earning	Earning	nation	Nation	trading	Trading
eat	Eating	number	Number	troops	Troop
ecology	Ecology	nurse	Nurse	valor	Valor
effort	Energy	object	Object	victory	Victory
egg	Egg	office	Office	virtuoso	Virtuoso
emergency	Emergency	operation	Operation	voyage	Voyage
entity	Entity	oracle	Oracle	warning	Warning
equality	Equality	organism	Organism	wealth	Wealth
evidence	Evidence	paper	Paper	wine	Wine
example	Example	party	Party	withdrawal	Withdrawal
eye	Eye	payment	Payment	wizard	Wizard
fauna	Fauna	people	People	woman	Woman
fear	Fear	percent	Percentage	woodland	Woodland
feline	Felinae	performance	Performance	world	World
film	Film	personnel	Human resources	year	Year
fingerprint	Fingerprint	place	Place	yen	Japanese yen
flight	Flight	plan	Plan	zoo	Zoo
flood	Flood	planning	Planning		

D. Czech Traveler Dataset

Table D.1.: All entities in the Czech Traveler dataset – Part 1. The 186 entities come from 103 annotations.

Full annotation	Noun phrase	Agreement	ANN 1	ANN 2
Adriatic sea near Vlore	Adriatic Sea	water	water	water
	Vlore	landscape	landscape	landscape
Ai-Petri Mountains	Ai-Petri Mountains	geological_formation	geological_formation	geological_formation
Albanian guide Kamil	Albanian guide Kamil	organism	organism	organism
Albanian Gypsies	Albanian gypsies	organism	organism	organism
Albanian riviera between Llogare and Qepara	Albanian riviera		landscape	sand
	Llogare			landscape
	Qeparo			landscape
Albanian souvenir T-shirts	Albanian souvenir T-shirts	artefact	artefact	artefact
Albanian Transport and Traffic	Albanian transport	vehicle	vehicle	vehicle
	traffic	vehicle	vehicle	vehicle
Almonds and Nuts	almonds	natural_object	natural_object	natural_object
	nuts	natural_object	natural_object	natural_object
Antiquities sold at local market	antiquities	artefact	artefact	artefact
	local market	structure	structure	structure
Archaeological site of Apolonia	archaeological site	structure	structure	structure
	Apolonia		landscape	artefact
Babele Natural Monument in Bucegi National Park	Babele Natural Monument		structure	geological_formation
	Bucegi National Park	vegetation	vegetation	vegetation
Baby bear at Ai-Petri Mountains	baby bear	organism	organism	organism
	Ai-Petri Mountains	geological_formation	geological_formation	geological_formation
Black house on Market Square	black house	structure	structure	structure
	Market Square	structure	structure	structure
Bran Castle	Bran Castle	structure	structure	structure
Bucegi National Park	Bucegi National Park	vegetation	vegetation	vegetation
Calanques de Piana	Calanques de Piana			geological_formation
Castle and Skanderbeg museum	castle	structure	structure	structure
	Skanderbeg Museum	structure	structure	structure
Cave City of Chufut-Kale near Bakhchisarai	cave city		landscape	geological_formation
	Chufut-Kale		structure	geological_formation
	Bakhchisarai			landscape
Ceahlau National Park	Ceahlau National Park	vegetation	vegetation	vegetation
Crimean Tatar's Camel at Ai-Petri Mountains	Crimean Tatar's camel	organism	organism	organism
	Ai-Petri Mountains	geological_formation	geological_formation	geological_formation
Divan Chamber of The Khan's Palace	Divan Chamber	structure	structure	structure
	Khan's Palace	structure	structure	structure
Drin river	Drin River	water	water	water
Duratoidea Waterfalls	Duratoidea Waterfalls	water	water	water
Equestrial statue of Skanderbeg	equestrial statue		structure	artefact
	Skanderbeg	organism	organism	organism
Ferry on Drin river	ferry	vehicle	vehicle	vehicle
	Drin River	water	water	water
Food stall of Crimean Tatars at Ai-Petri Mountains	food stall	structure	structure	structure
	Crimean Tatars	organism	organism	organism
	Ai-Petri Mountains	geological_formation	geological_formation	geological_formation
Grand Canyon around Auzun-Uzen River	grand canyon	geological_formation	geological_formation	geological_formation
	Auzun-Uzen River	water	water	water
Holy Monastery of Durau	Holy Monastery	structure	structure	structure
	Durau	landscape	landscape	landscape
Holy Assumption Bakhchisarai Monastery	Holy Assumption Bakhchisarai Monastery	structure	structure	structure
Jetee du Dragon	Jetee du Dragon			
Kinoteatr cinema	Kinoteatr cinema	structure	structure	structure
Kvas sale	kvas sale	artefact	artefact	artefact
Landscape around GR20 between Monte d'Oro and Les Cascades des Anglais	landscape	landscape	landscape	landscape
	GR20		geological_formation	structure
	Monte d oro			geological_formation
	Les Cascades des Anglais			water
Landscape around Lacul Balea	landscape	landscape	landscape	landscape
	Lacul Balea			water
Landscape between Dhermi and Qeparo	Dhermi			landscape
	landscape	landscape	landscape	landscape
	Qeparo			landscape
Landscape between Gjirokaster and Korce	landscape	landscape	landscape	landscape
	Gjirokaster	landscape	landscape	landscape
	Korce	landscape	landscape	landscape
Landscape between Sarande and Gjirokaster	landscape	landscape	landscape	landscape
	Sarande	landscape	landscape	landscape
	Gjirokaster	landscape	landscape	landscape
Landscape near Gjirokaster	landscape	landscape	landscape	landscape
	Gjirokaster	landscape	landscape	landscape
Mosque	mosque	structure	structure	structure
Mountains in Central Corsica	mountains	geological_formation	geological_formation	geological_formation
	Central Corsica	landscape	landscape	landscape

Table D.2.: All entities in the Czech Traveler dataset – Part 2

Museum at Market Square	museum	structure	structure	structure
	Market Square	structure	structure	structure
Napoleon's monument at Place St Nicolas	Napoleons monument		structure	artefact
	Place St Nicolas	structure	structure	structure
Neamt Monastery	Neamt Monastery	structure	structure	structure
Old bazar / market	old bazar		structure	
	market		structure	
On the seashore	seashore		landscape	sand
Orthodox Priest from Neamt Monastery	orthodox priest	organism	organism	organism
	Neamt Monastery	structure	structure	structure
Park surrounding Livadia Palace	park	vegetation	vegetation	vegetation
	Livadia Palace	structure	structure	structure
Peles Castle	Peles Castle	structure	structure	structure
Ruins	ruins	structure	structure	structure
Ruins of Dracula's Castle in Arefu	ruins	structure	structure	structure
	Dracula's Castle	structure	structure	structure
	Arefu	structure	structure	structure
Russian Fleet in South Bay	Russian Fleet	vehicle	vehicle	vehicle
	South Bay	water	water	water
Russian pensioners	Russian pensioners	organism	organism	organism
Sheep in Bucegi National Park	sheep	organism	organism	organism
	Bucegi National Park		landscape	vegetation
Statue of Lenin and Red Square stall at Lenin's Square	statue	structure	structure	structure
	Lenin	organism	organism	organism
	Red Square	structure	structure	structure
	Lenin's Square	structure	structure	structure
Swallows's Nest	Swallow's Nest		structure	natural_object
Syri i Kalter / Blue Eye Spring	Syri i Kalter			
	Blue Eye Spring	water	water	water
The Great Basilica	Great Basilica	structure	structure	structure
The Khan's Palace	Khan's Palace	structure	structure	structure
Two generations of Romanian women in traditional clothes	two generations	organism	organism	organism
	Romanian women	organism	organism	organism
	traditional clothes	artefact	artefact	artefact
Typical architecture in UNESCO World Heritage City of Berat	typical architecture	structure	structure	structure
	UNESCO World Heritage city	landscape	landscape	landscape
	Berat	landscape	landscape	landscape
Vorontsov Alupka Palace Museum	Vorontsov Alupka Palace Museum		artefact	structure
Wine Tasting in Massandra	wine tasting			event
	Massandra	landscape	landscape	landscape
Yalta waterfront	Yalta waterfront	water	water	water
The Merry Cemetery protected by UNESCO	Merry Cemetery	structure	structure	structure
	UNESCO			
Shepherd in Bucegi National Park	Shepherd	organism	organism	organism
	Bucegi National Park	vegetation	vegetation	vegetation
Curtea de Arges	Curtea de Arges			
Opera house	Opera house	structure	structure	structure
Young people meeting near statue of Lenin	people	organism	organism	organism
	statue		structure	artefact
	Lenin	organism	organism	organism
Crimean Parliament	Crimean Parliament	structure	structure	structure
Ukainian girls chatting in front of fountain	Ukrainian girls	organism	organism	organism
	fountain		structure	artefact
Tank as Monument to freedom	Tank	vehicle	vehicle	vehicle
	Monument		structure	artefact
Church on the rock in Foros	church	structure	structure	structure
	rock		landscape	geological_formation
	Foros	landscape	landscape	landscape
Sevastopol	Sevastopol	landscape	landscape	landscape
Sleeping dogs in Sevastopol	sleeping dogs	organism	organism	organism
	Sevastopol	landscape	landscape	landscape
Sailors from Sevastopol	Sailors	organism	organism	organism
	Sevastopol	landscape	landscape	landscape
Small business in Sevastopol	small business	structure	structure	structure
	Sevastopol	landscape	landscape	landscape
Sea lions in Sevastopol's delphinarium	Sea lions	organism	organism	organism
	Sevastopol's delphinarium	structure	structure	structure
Dolphin in Sevastopols delphinarium	Dolphin	organism	organism	organism
	delphinarium	structure	structure	structure
St Vladimir Cathedral in Chersonesus near Sevastopol	St Vladimir Cathedral	structure	structure	structure
	Chersonesus	landscape	landscape	landscape
	Sevastopol	landscape	landscape	landscape
Khersones near Sevastopol	Khersones	landscape	landscape	landscape
	Sevastopol	landscape	landscape	landscape

Table D.3.: All entities in the Czech Traveler dataset – Part 3

Greek ruins in Khersones near Sevastopol	Greek ruins	structure	structure	structure
	Khersones	landscape	landscape	landscape
	Sevastopol	landscape	landscape	landscape
St Vladimir Cathedral in Khersones near Sevastopol	St Vladimir Cathedral	structure	structure	structure
	Khersones	landscape	landscape	landscape
	Sevastopol	landscape	landscape	landscape
Kalamita fortress in Inkerman	Kalamita fortress	structure	structure	structure
	Inkerman	landscape	landscape	landscape
Rock and cave town of Eski Kermen	Rock and cave town	landscape	landscape	landscape
	Eski Kermen	landscape	landscape	landscape
Artifical palm trees and beach with signposts	artificial palm trees		vegetation	landscape
	beach		natural_object	sand
	signposts	artefact	artefact	artefact
Countryside around Jablonica	countryside	landscape	landscape	landscape
	Jablonica	landscape	landscape	landscape
Plansarsko lake	Plansarsko Lake	water	water	water
Landscape around Jezersko	landscape		natural_object	landscape
	Jezersko	landscape	landscape	landscape
Logarska dolina	Logarska Dolina		geological_formation	landscape
Velika planina	Velika Planina		vegetation	landscape
Church in Jezersko, Grintovec mountain at background	church	structure	structure	structure
	Jezersko	landscape	landscape	landscape
	Grintovec Mountain	geological_formation	geological_formation	geological_formation
Karst Cave Vilenica	Karst Cave Vilenica		geological_formation	geological_formation
Predjama Castle	Predjama Castle	structure	structure	structure
Lipizzaner horse in Lipica Stud farm	Lipizzaner horse	organism	organism	organism
	Lipica Stud farm		structure	landscape
Neoclassical Palace in Buje	Neoclassical Palace	structure	structure	structure
	Buje	landscape	landscape	landscape
Motovun	Motovun	landscape	landscape	landscape
Istarske Toplice	Istarske Toplice	landscape	landscape	landscape
Belfry	belfry		structure	
Main Town Gate	main town gate	structure	structure	structure
Monument of Glagolitic Alley	Monument		structure	artefact
	Glagolitic Alley		vegetation	artefact
Children in swimming pool	children	organism	organism	organism
	swimming pool		structure	artefact
Oprtalj	Oprtalj	landscape	landscape	landscape
Umago	Umago	landscape	landscape	landscape

Table D.4.: Entities with inter-annotator agreement in the Czech Traveler dataset – Part 1: list of the 143 entities classified to nine classes with inter-annotator agreement. This dataset was used in the SCM and BOA experiments. Column **NE** indicates, if the entity is considered a named entity, column **THD** indicates if THD was used for WordNet mapping. Column **WordNet** lists the WordNet mapping. Column **Wikipedia direct** gives the first article Wikipedia search returned for the noun phrase. Column **WordNet to Wikipedia** gives the first Wikipedia article returned for the entry in the WordNet column.

#	Noun_phrase	NE	Agreement	THD	WordNet	Wikipedia direct	WordNet to Wikipedia
1	Adriatic Sea	1	water		Adriatic Sea	Adriatic Sea	Adriatic Sea
2	Vlore	1	landscape	1	shore lines	Vlorë	Shore
3	Ai-Petri Mountains	1	geological formation		mountain	Crimean Mountains	Mountain
4	Albanian guide Kamil	1	organism	1	statesman	Egyptians	Statesman
5	Albanian gypsies		organism		gypsy	Romani people	Gypsy
6	Albanian souvenir T-shirts		artefact		shirt	Hair (musical)	Shirt
7	Albanian transport		vehicle		transport	Transport in Albania	Transport
8	traffic		vehicle		traffic	Traffic	Traffic
9	almonds		natural object		almond	Almond	Almond
10	nuts		natural object		nut	Nomenclature of Territorial Units for Statistics	Nut
11	antiquities		artefact		antiquity	Antiquities	Antiquity
12	local market		structure		market	Media market	Market
13	archaeological site		structure		site	Archaeological site	Site
14	Bucegi National Park	1	vegetation		park	Protected areas of Romania	Park
15	baby bear		organism		bear	The Bear family	Bear
16	Ai-Petri Mountains	1	geological formation		mountain	Crimean Mountains	Mountain
17	black house		structure		house	Black house	House
18	Market Square	1	structure		market square	Market square	Market square
19	Bran Castle	1	structure		castle	Bran Castle	Castle
20	Bucegi National Park	1	vegetation		park	Protected areas of Romania	Park
21	castle		structure		castle	Castle	Castle
22	Skanderbeg Museum	1	structure		museum	Skanderbeg Museum	Museum
23	Ceahlau National Park	1	vegetation		park	Ceahlău Massif	Park
24	Crimean Tatar's camel		organism		camel	Nogai Horde	Camel
25	Ai-Petri Mountains	1	geological formation		mountain	Crimean Mountains	Mountain
26	Divan Chamber	1	structure		chamber	Topkapı Palace	Chamber
27	Khan's Palace	1	structure		palace	Bakhchisaray Palace	Palace
28	Drin River	1	water		river	Drin (river)	River
29	Duratoidea Waterfalls	1	water		waterfall		Waterfall
30	Skanderbeg	1	organism	1		Skanderbeg	Skanderbeg
31	ferry		vehicle		ferry	Ferry	Ferry
32	Drin River	1	water		river	Drin (river)	River
33	food stall		structure		stall	Street food	Stall
34	Crimean Tatars		organism		Tatar	Crimean Tatars	Tatar
35	Ai-Petri Mountains	1	geological formation		mountain	Crimean Mountains	Mountain
36	grand canyon		geological formation		Grand Canyon	Grand Canyon	Grand Canyon
37	Auzun-Uzen River	1	water		river		River
38	Holy Monastery	1	structure		monastery	Saint Archangels Monastery	Monastery
39	Durau	1	landscape	1	resort	Durău	Resort
40	Holy Assumption Bakhchisarai Monastery	1	structure		monastery		Monastery
41	Kinoteatr cinema		structure		cinema	Metropolitan Association of Upper Silesia	Cinema
42	kvas sale		artefact		sale	Kvass	Sale
43	landscape		landscape		landscape	Landscape	Landscape
44	landscape		landscape		landscape	Landscape	Landscape
45	landscape		landscape		landscape	Landscape	Landscape
46	landscape		landscape		landscape	Landscape	Landscape
47	Gjirokaster	1	landscape	1	city	Gjirokastër	City
48	Korce	1	landscape	1	city	Korçë	City
49	landscape		landscape		landscape	Landscape	Landscape
50	Sarande	1	landscape	1	capital	Sarandë	Capital

Table D.5.: Entities with inter-annotator agreement in the Czech Traveler dataset – Part 2

#	Noun_phrase	NE	Agreement	THD	WordNet	Wikipedia direct	WordNet to Wikipedia
51	Gjirokaster	1	landscape	1	city	Gjirokastër	City
52	landscape		landscape		landscape	Landscape	Landscape
53	Gjirokaster	1	landscape	1	city	Gjirokastër	City
54	mosque		structure		mosque	Mosque	Mosque
55	mountains		geological formation		mountain	Mountain	Mountain
56	Central Corsica	1	landscape		corsica	Ajaccio	Corsica
57	museum		structure		museum	Museum	Museum
58	Market Square	1	structure		market square	Market square	Market square
59	Place St Nicolas	1	structure	1		Haapsalu Castle	Haapsalu Castle
60	Neamt Monastery	1	structure		monastery	Neamț Monastery	Monastery
61	orthodox priest		organism		priest	List of children of clergy	Priest
62	Neamt Monastery	1	structure		monastery	Neamț Monastery	Monastery
63	park		vegetation		park	Park	Park
64	Livadia Palace	1	structure		palace	Livadia Palace	Palace
65	Peles Castle	1	structure		castle	Peleș Castle	Castle
66	ruins		structure		ruin	Ruins	Ruins
67	ruins		structure		ruin	Ruins	Ruins
68	Dracula's Castle	1	structure		castle	Dracula's Castle	Castle
69	Arefu	1	structure	1	commune	Arefu	Commune
70	Russian Fleet	1	vehicle		fleet	Russian Empire	Fleet
71	South Bay	1	water		bay	South Bay	Bay
72	Russian pensioners		organism		pensioner	Russian Pensioners' Party	Pensioner
73	sheep		organism		sheep	Domestic sheep	Sheep
74	statue		structure		statue	Statue	Statue
75	Lenin	1	organism		Lenin	Vladimir Lenin	Vladimir Lenin
76	Red Square	1	structure		square	Red Square	Square
77	Lenin's Square	1	structure		square	Freedom Square, Tbilisi	Square
78	Blue Eye Spring	1	water		spring	Blue Eyed Six	Spring
79	Great Basilica	1	structure		Basilica	Great Basilica of Pliska	Basilica
80	Khan's Palace	1	structure		palace	Bakhchisaray Palace	Palace
81	two generations		organism		generation	Transformers: Generation 2	Generation
82	Romanian women		organism		woman	Romania women's national handball team	Woman
83	traditional clothes		artefact		clothes	Vietnamese clothing	Clothing
84	typical architecture		structure		architecture	Shinmei-zukuri	Architecture
85	UNESCO World Heritage city		landscape		city	City	City
86	Berat	1	landscape	1	town	Berat	Town
87	Massandra	1	landscape	1	asteroid	Massandra	Asteroid
88	Yalta waterfront		water		waterfront	Trolleybus	Waterfront
89	Merry Cemetery	1	structure		cemetery	Merry Cemetery	Cemetery
90	Shepherd		organism		shepherd	Shepherd	Shepherd
91	Bucegi National Park	1	vegetation		park	Protected areas of Romania	Park
92	Opera house		structure		opera house	Opera house	Opera house
93	people		organism		people	People	People
94	Lenin	1	organism		Lenin	Vladimir Lenin	Vladimir Lenin
95	Crimean Parliament		structure		parliament	Supreme Council of Crimea	Parliament
96	Ukrainian girls		organism		girl	Girl	Girl
97	Tank		vehicle		tank	Tank	Tank
98	church		structure		church	Church	Church
99	Foros	1	landscape	1	town	Foros	Town
100	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol

Table D.6.: Entities with inter-annotator agreement in the Czech Traveler dataset – Part 3

#	Noun_phrase	NE	Agreement	THD	WordNet	Wikipedia direct	WordNet to Wikipedia
101	sleeping dogs		organism		dog	Dog	Dog
102	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
103	Sailors		organism		sailor	Sailor	Sailor
104	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
105	small business		structure		business	Business	Business
106	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
107	Sea lions		organism		sea lion	Sea lion	Sea lion
108	Sevastopol's delphinarium		structure	1	aquarium	Sevastopol	Aquarium
109	Dolphin		organism		dolphin	Dolphin	Dolphin
110	delphinarium		structure	1	aquarium	Dolphinarium	Aquarium
111	St Vladimir Cathedral	1	structure		cathedral	St Volodymyr's Cathedral	Cathedral
112	Chersonesus	1	landscape	1	harbor	Chersonesus	Harbor
113	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
114	Khersones	1	landscape	1	ship	Khersones	Ship
115	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
116	Greek ruins		structure		ruin	Magna Graecia	Ruins
117	Khersones	1	landscape	1	ship	Khersones	Ship
118	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
119	St Vladimir Cathedral	1	structure		cathedral	St Volodymyr's Cathedral	Cathedral
120	Khersones	1	landscape	1	ship	Khersones	Ship
121	Sevastopol	1	landscape		Sevastopol	Sevastopol	Sevastopol
122	Kalamita fortress		structure		fortress	Fortification	Fortress
123	Inkerman	1	landscape	1	town	Inkerman	Town
124	Rock and cave town		landscape		town	Rock Cave, West Virginia	Town
125	Eski Kermen	1	landscape	1		Dugout (shelter)	Dugout (shelter)
126	signposts		artefact		signpost	Traffic sign	Traffic Sign
127	countryside		landscape		countryside	Rural area	Rural area
128	Jablonica	1	landscape	1	village	Jablonica	Village
129	Plansarsko Lake		water		lake	Lake	Lake
130	Jezersko	1	landscape	1	municipality	Jezersko	Municipality
131	church		structure		church	Church	Church
132	Jezersko	1	landscape	1	municipality	Jezersko	Municipality
133	Grintovec Mountain	1	geological formation		mountain	Grintovec	Mountain
134	Predjama Castle	1	structure		castle	Predjama Castle	Castle
135	Lipizzaner horse		organism		horse	Lipizzan	Horse
136	Neoclassical Palace		structure		palace	Neoclassicism	Palace
137	Buje	1	landscape	1	town	Buje	Town
138	Motovun	1	landscape	1	village	Motovun	Village
139	Istarske Toplice	1	landscape	1		Istarske Toplice	Istarske Toplice
140	main town gate		structure		gate	Faust: The First Part of the Tragedy	Gate
141	children		organism		child	Child	Child
142	Oprtalj	1	landscape	1	community	Oprtalj	Community
143	Umago	1	landscape	1	city	Umag	City