Extracting Semantic Information for e-Commerce

Bruno Charron, Yu Hirate, David Purcell, and Martin Rezk(✉)

Rakuten Inc., Tokyo, Japan
{bruno.charron,yu.hirate,david.purcell,martin.rezk}@rakuten.com

Abstract. Rakuten Ichiba uses a taxonomy to organize the items it sells. Currently, the taxonomy classes that are relevant in terms of profit generation and difficulty of exploration are being manually extended with data properties deemed helpful to create pages that improve the user search experience and ultimately the conversion rate. In this paper we present a scalable approach that aims to automate this process, automatically selecting the relevant and semantically homogenous subtrees in the taxonomy, extracting from semi-structured text in items descriptions a core set of properties and a popular subset of their ranges, then extending the covered range using relational similarities in free text. Additionally, our process automatically tags the items with the new semantic information and exposes them as RDF triples. We present a set of experiments showing the effectiveness of our approach in this business context.

1 Introduction

Semantic technologies are not new in the e-commerce business. In particular, ontologies (or ontology-like artifacts) have been used for a number of different tasks that go from data integration [11] to items classification and search support [7]. Rakuten Ichiba (the largest e-commerce site in Japan) uses a large legacy taxonomy with around 40,000 classes to organize the items and provide the users with discovery axes relevant to their searches. In order to assist the user with her search, the catalog team is manually extending the taxonomy and the items with new semantic information. They start by selecting the classes in this taxonomy that need to be improved based on profitability and difficulty of exploration. For each of these classes, they study the domain together with the customers shopping behavior, such as search keywords, browsed items, etc. Based on that, they extend the taxonomy with a small set of properties deemed helpful to the customers along with the most representative values (not necessarily all) in the range of these properties. Finally, they use these properties/values to create pages that help the users to explore the items in the class.

Being completely manual, this taxonomy extension effort currently requires a massive amount of time and human effort. Acquiring domain knowledge and modeling the properties alone for a single class can take days, the operation is error-prone, and the result cannot be easily ported to other Rakuten market places across the globe (Taiwan, France, US, Germany, etc.). As such, there is

© Springer International Publishing AG 2016
P. Groth et al. (Eds.): ISWC 2016, Part II, LNCS 9982, pp. 273–290, 2016.
DOI: 10.1007/978-3-319-46547-0_27

an important business value in automating this process, both to speed up the completion of the project and to drastically reduce the cost of this task.

This work raises a number of questions: (i) how to automatically select relevant and semantically homogenous subtrees in the taxonomy which would most benefit from being extended with data properties; (ii) how to extract a core set of properties from existing datasets that can effectively assist the users in their product searches; (iii) how to extract a representative set of range values to be displayed together with these properties; (iv) how to find for each item in each class the corresponding property values; (v) how to provide a generalizable solution (to arbitrary taxonomies, languages, and datasets) that scales to run over tens of thousands of classes and TB of data.

In this paper we propose answers to these questions adopting and extending techniques from ontology-based information extraction and triple extraction, and proposing new techniques when needed. We present an end-to-end scalable unsupervised approach that automatizes the process of extracting semantic information from item descriptions and shopping behaviors to improve the user experience. This process adds new properties to the relevant subtrees of the taxonomy and tags the items with this new information, exposing the outcome as RDF triples. The pipeline of our approach is depicted in Fig. 1. In this work when we say taxonomy/ontology we do not necessary mean OWL ones, and when we say triples do not necessary mean RDF ones. However, we plan to use this project to push for the use of these semantic web standards inside Rakuten.

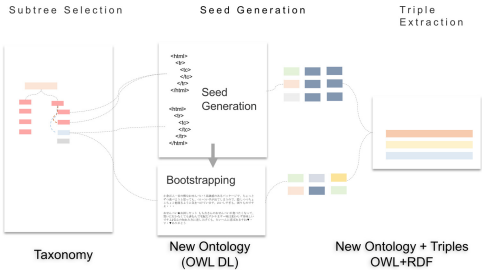


Fig. 1. Expected outcome

Our main contributions can be summarized as follows:

- A novel technique to select relevant subtrees in the taxonomy in terms of overall profitability, ease of product discovery and similarity of users' shopping behavior.
- An extension to the work presented in [23] to extract, for the relevant subtrees, an initial set (seed) of popular property-value list from semi-structured text. The extension includes a new technique for the aggregation of synonymous property names, as well as an improvement in the precision of the result by exploiting users' shopping data.

– An approach, based on neural networks [16], to extend the initial seed with new relevant values using semantic similarity.
– A set of experiments to measure the precision, the coverage, and the scalability of these techniques.

The remainder of the paper is organized as follows: Sect. 2 describes the business need, and highlights the problems with the current approach. In Sect. 3 we briefly survey other related works. In Sect. 4 we describe our approach towards data property extraction, range extraction, and triple generation. Section 5 gives an overview of the architecture of our current implementation of the proposed approach. In Sect. 6 we provide a number of experiments evaluating the correctness and scalability of our framework. Section 7 concludes the paper.

2 Business Case

Rakuten Group is one of the world's leading Internet service companies, providing a variety of consumer and business-focused services including e-commerce, eBooks, travel, banking, securities, credit card, e-money, portal and media, online marketing, professional sports, etc. Currently, around 40[1] of the total operating income comes from Rakuten Ichiba, the leading e-commerce website in Japan, which is Rakuten Group's online shopping mall where third-party merchants can set up shops and sell their products. Rakuten Ichiba offers around 200 million items classified in a large legacy taxonomy (there are only classes and subclass relations between them) with around 40,000 classes. The first version of this taxonomy was developed around 2001 (three years before the standardization of OWL) and has been evolving since then, although it was never moved to a well-known ontology language. Currently there are several internal projects to shrink and improve the quality of this taxonomy, aiming to enhance the user experience, increase the conversion rate[2], and as a consequence increase the *gross merchandise sales* (abbr. GMS) per class. In this work we focus on the project that aims to extend the relevant taxonomy classes and the items with new semantic information deemed helpful to the user.

An appeal of Rakuten Ichiba which contributed to its initial popularity is the ease and freedom with which the merchants can register products on the platform, allowing them to freely design their item descriptions in order to construct a shop identity and a connection with the customers. As a result, any information about the items must be found in their titles and descriptions specified by the merchants, that is, HTML code mostly consisting of free-form text, semi-structured text (table-like free-form text), structured text (tables) and images. This data is therefore the primary source to extract item-specific information in this project but it is also used to automatically acquire domain-specific knowledge as it reflects the knowledge of the merchants.

[1] http://global.rakuten.com/corp/investors/documents/results/.
[2] Proportion of customers making a purchase within a given browsing session.

When extending the taxonomy, it is necessary to avoid extracting properties for *every* single class. First, because of the legacy nature of the taxonomy, some classes are either obsolete, or in an unintuitive branch, or too specific to be interesting, therefore displaying data properties for those classes will not improve the user experience (abbr. UX). Recall that here we are not trying to build a complete model of the domain, like the work in [17], but use the ontology to improve the UX. Second, extending the whole taxonomy would not only dramatically increase its already massive size but also lead to unintuitive results if we extract properties of semantically inhomogeneous classes. For instance, the class Wine has among its subclasses Red Wine, White Wine, and Wine Accessories. Clearly the properties for Wine and Wine Accessories are very different. Thus, trying to extract property-values from *all* the items in the Wine category directly leads to unintuitive (although correct) results, such as "yellow" for the property "color".

Currently, the extension of the taxonomy is done manually and we aim to reduce the human effort needed for this task, although some human work might still be needed. A production-level system should have a precision above 80 %, and the cost of the deployment and maintenance of the proposed solution should not exceed the benefits from the use of the system.

3 Related Work

This work intersects mainly with two fields: Ontology-based information extraction and triple extraction. In this section we will briefly survey some of the works in these areas and their relation with our approach. It is worth noting that none of the articles mentioned here tackle the problem of selecting subtrees as described in Sects. 4 and 5.

Several studies mentioned in this section rely heavily on a number of linguistic resources (such as Yago, DBpedia, Wikipedia) and complex NLP tools (such as entity/predicate disambiguation and matching [24]). These resources and tools for languages different than English (we work with Japanese texts) are often not as accurate, and the inaccuracies tend to accumulate and propagate through layers. In addition, performing such analysis over terabytes of data can be computationally expensive. Thus, in this work we only use a tokenizer to prepare the datasets and we use data mining and machine learning to extract relations and triples. In the future we plan to use NLP techniques to improve the results when our techniques perform sub-optimally.

OBIE: In Ontology-based information extraction (abbr. OBIE) [25] the ontology is used to drive the information extraction process. In [21], the authors describe an OBIE application (GATE) for e-business. In this approach, the properties are already given and the concepts are selected manually with the help of domain experts. The work in [1] extracts properties names using an NLP approach, but the extraction process relies on a predefined (manually by experts) semantic lexicon, and on a manually created set of rules. Another NLP approach for OBIE

is the one in [2], where the extracted relations have to be mapped to an existing (manually created) list of properties. [22] presents a system called RelExt for relation extraction. They use semi-structured text to map terms to concepts, and linguistic analysis to automatically find relations between those terms. In addition, they use a number of statistical metrics to score the popularity of the triples. [19] presents a system called DOGMA for relation extraction from free text. This system first finds verbs-objects in the text, then builds semantics sets by clustering the verbs and the terms, and finally finds the relations between those sets of terms. Unlike our work (and [22]) they make no use of semi-structure data, but the precision reported is rather low (around 11 %). [9] introduces *opal*, an ontology-based web pattern analysis approach that extracts semantic information (such as properties, values, and classes) from web forms exploiting a number of prolog rules. The classification step (that maps terms to concepts) requires terms in the domain to be annotated to create a set of ground facts, such as, City(London). [13] tackles the same problem but instead of having a logic-based approach, they use a machine learning approach (they use Bayesian classifiers). Some manual labor is also required to build the training sets of the classifiers. [17] presents the never-ending learning paradigm for machine learning, that endlessly extends a small original ontology (Tbox) with new classes, properties, and facts (Abox) from the web.

It is worth noticing that in the works mentioned above, the authors start from text, and then they populate the classes/relation in the ontology. In our scenario, we start from the ontology, and look for its instances in text.

We also include under the OBIE discussion the extraction of attributes and values such as the works presented in [4,6,12,14,23]. Although they do not constitute an ontology, the same techniques can be applied to extract ontological properties (in fact, that is what we do). These studies propose similar unsupervised frameworks that extract a set of properties and values from HTML semi-structure data. After this core set has been extracted, in [4,23], they use machine learning (Conditional Random Fields) to expand the set of values by extracting values from free text. [4] takes into account the popularity of the properties, as we do, but they use user reviews instead of user's queries. [12] extracts first an initial seed of property-values from user queries, and then it uses distant supervision to extend it. [23,26] also present methods to aggregate semantically equivalent property names, in Sect. 5 we discuss the differences with our approach. [18] uses supervised learning to classify HTML fragments, and then combines two techniques to extract the property-values: the first strategy is similar to the one in [23], and the second one is based on the outcome of a number of noisy annotators that rely on manually labeled data.

Triple Extraction from Text: Works in this field aim to detect semantic relations between instances in the text. [24] presents two complementary approaches to extract relations: a rule-based approach and a machine learning approach. Their rule-based approach provides high precision but it requires manual work to build the rules. The ML based approach relies heavily on a number of NLP tools. The systems presented in [8,14] follow a similar approach to extract triples from

unstructured text. In particular, [14] is a plugin for [21]. Observe that their goal, in all the cases above, is different from ours. They aim to discover as many relations as they can (+7000 in [24]). Here we aim to extract a small set of "helpful" properties for and from an automatically selected taxonomy subtree.

4 Overview of the Solution

In this section we describe the different steps of our approach. These steps are depicted in Fig. 2. Observe that this solution is general, and it does not depend on any Rakuten-specific artifact, therefore in this section we only explain *what* each step should do. In Sect. 5 we describe *how* we implemented each of these steps in the case of Rakuten Ichiba. Also note that the production of the formal OWL ontology and RDF triples is not mandatory in this process. Although in our implementation we do output the ontology in this format (c.f. Sect. 5.2), the adoption of W3C standards for these artifacts in Rakuten is still under consideration, and this work represents the first steps towards it. We will make use of Description Logics (DL) terminology, but if the reader is not familiar with this formalism s/he can skip these parts.

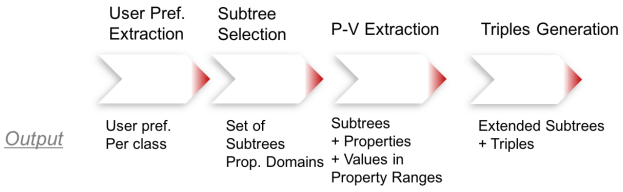


Fig. 2. Extracting semantic information to extend an ontology

User Preference Extraction: Since the end goal is to assist the users in their purchases, the first step is to extract relevant information about their shopping behavior. What data is needed here depends on the particular implementation of the following steps, but query logs and browsed items are some examples of the type of data that can be used.

Subtree Selection: Intuitively, relevant subtrees are those that: *(i)* have *high business impact*; *(ii)* can benefit from *new discovery axes*, i.e. from a UI perspective, they need some navigational assistance; and *(iii)* are *semantically homogeneous* from the user perspective. A class has a high need for navigational assistance (NNA) if it is "sufficiently far" from purchases, that is, the purchases occur in one of its non-immediate subclasses, or the diversity of the class in terms of the number of popular items is high. A class has a high business impact (BI) if the items in the class contribute significantly to the GMS. For instance, Water is a class with high BI but with low NNA because it contains few popular items

which amount to a large sales volume. On the other hand, Red Wine has both high BI and high NNA because the large sales volume of the class is distributed over a high number of moderately popular items. A subtree of the taxonomy is semantically homogeneous from the user perspective (UH) if users associate the classes in the subtree with similar terms. For instance, although American whisky and wine are semantically similar (alcoholic drinks), these classes are reached in different ways. Most searches leading to American whisky (in Japan) are very specific, such as "Old Crow"; on the other hand, searches leading to red wine, white wine, and even Champagne tend to be more general and often overlap, for instance "French wine" (non-alcoholic wine would be in a different subtree). Observe that this way of splitting the tree does not only consider the semantics of the class, but also the behavior of the shoppers. In our example, shoppers buying American whisky will tend to have (statistically) a better background in that domain, while shoppers buying wine are more diverse. This process is illustrated in Fig. 3.

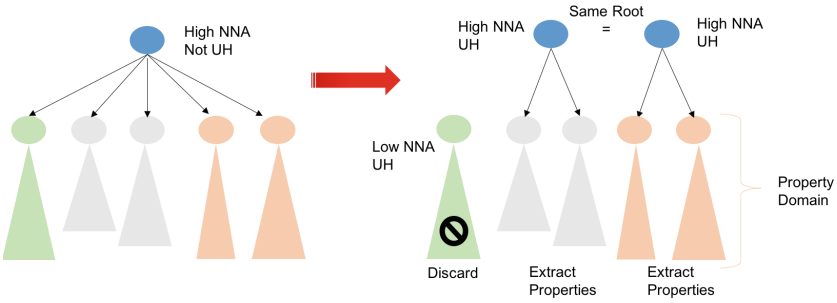


Fig. 3. Subtree selection process

Property-Value Extraction: This step consists of extending a subtree, T, selected in the previous step, with a *relevant* set of data property. The definition of relevant depends on the particular goal, but in any case it must align with the requirements of the particular use-case. The domain of these new properties is the union of the classes in $T \setminus root(T)$, that is, all the classes in T minus the root. Formally, let C be the root of the tree representing the subclass relation, $D_1 \dots D_n$ the subclasses of C in T, and R a new relation, then the ontology (Tbox) is extended with the following OWL 2 DL axiom (in DL notation for succinctness): $\exists R \equiv D_1 \sqcup \dots \sqcup D_n$.

Once the ontology for the subtree T has been extended with the set of relevant properties $R_1 \dots R_n$, the next step is to find a fragment of the range of these properties (literals) that is popular among the users that browse the tree T, and that will be used to generate the triples. In this work we focus on data properties, but this work can be extended to object properties using the approaches described in [9,13,17].

Triples Generation: The final step is to link the items in T with the properties and values which have been extracted. Observe that depending on the scenario, a confidence score might be attached to each triple (s, p, o) indicating how certain the system is that the item s has the property p with value v. In such case, one would need to reify the tuples using the standard technique [3].

5 Architecture

In Sect. 4 we described the general flow of the approach. In this section we present how we implemented the different steps in Rakuten. The architecture, which is illustrated in Fig. 4, can be divided in four different layers: (i) the inputs, i.e., artifacts such as the Rakuten taxonomy, the search logs, etc.; (ii) the computational framework on which we run the different modules of the system; (iii) the processing layer, in charge of generating the intermediate artifacts from the input: sets of subtrees, tokenized item descriptions, word2vec models, etc.; (iv) the extension layer, in charge of extending the taxonomy using the intermediate artifacts and exposing the output as RDF. First we briefly describe the input layer and the computational framework and then we give closer looks to the processing and extension layers.

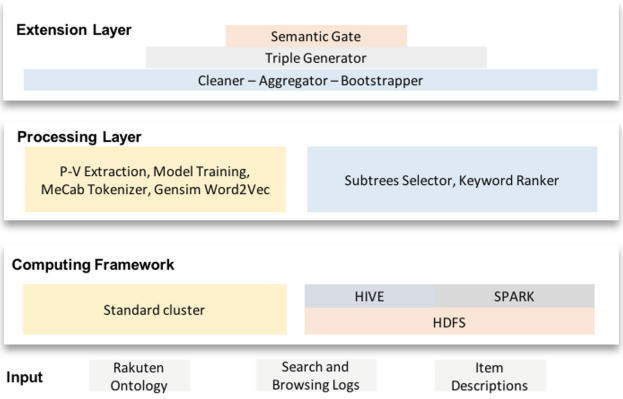


Fig. 4. System architecture

Input Layer: The system takes the following artifacts as input:

– *Rakuten Taxonomy:* The Rakuten taxonomy contains 38,167 classes: 35 with depth 1, 405 with depth 2, 3,790 with depth 3, 15,849 with depth 4, and the remaining classes with depth 5. Each class has exactly one parent except for the root. Unfortunately it is not formalized in a well-known ontology language. The intended semantics behind the parent-child relation is the same one as `owl:subclass`.

- *Search and Browsing Logs:* This dataset (2TB per year) contains information about all the search queries executed by users of Rakuten Ichiba: keywords, class being browsed, items browsed after the query, etc.
- *Item Descriptions:* This dataset (800GB per year) contains the HTML pages describing each item sold in Rakuten Ichiba, together with the class in which they belong and other metadata irrelevant for this project.

Computing Framework: Given the size of the datasets, the system relies on distributed computing frameworks to perform computations at the level of the full taxonomy, i.e. computations which require considering all the classes. A Hive cluster with 1000 nodes is used to pre-process and analyze the query logs then score the search keywords. A Spark cluster with 200 nodes is used to select the subtrees for which the taxonomy needs to be extended. On the other hand, the processing of the individual subtrees is independent and bounded in memory and computational requirements for each instance. Therefore, the subtree-level pipeline, which consists of a series of Python scripts, is run on a standard cluster and does not require a particular computing framework to scale.

5.1 Processing Layer

Subtree Selector. This modules selects the subtrees in the Rakuten taxonomy that will be extended. Next we give concrete definition of the different abstract concepts listed in Sect. 4.

Need for Navigational Assistance: To measure the need for new discovery axes of a subtree, we compute its GMS *diversity*, defined as $\exp(-\sum_i p_i \ln p_i)$ where the sum is over the items in the subtree and p_i is the proportion of the total GMS of the subtree which is due to the item i. This is the exponential of the Shannon entropy of the subtree's item-level GMS which intuitively represents the effective number of items in a subtree making up its GMS. A subtree is said to have a high need for navigational assistance (NNA) if its effective number of items is more than $Z_1 = 2^{15}$. It is said to have a low NNA, and is therefore discarded, if its effective number of items is less than $Z_2 = 2^7$. These values for Z_1 and Z_2 are based on an initial exploration of the data and the final values will be decided by the catalog team.

Business Impact: The business impact is not used in addition to the NNA requirement as we found in practice that subtrees not discarded by the previous requirement have high enough business impact. Indeed, a counter-example would necessitate a large number of items with very low and almost evenly distributed sales volumes, which is not found in our datasets.

Semantic Homogeneity: We use search query logs to measure how semantically homogeneous a given subtree t_1 is. For each node with depth 1 in t_1, we compute the set of search keywords leading to that node (class). A keyword is said to lead to a class if a user searching for this keyword clicked on an item of this class

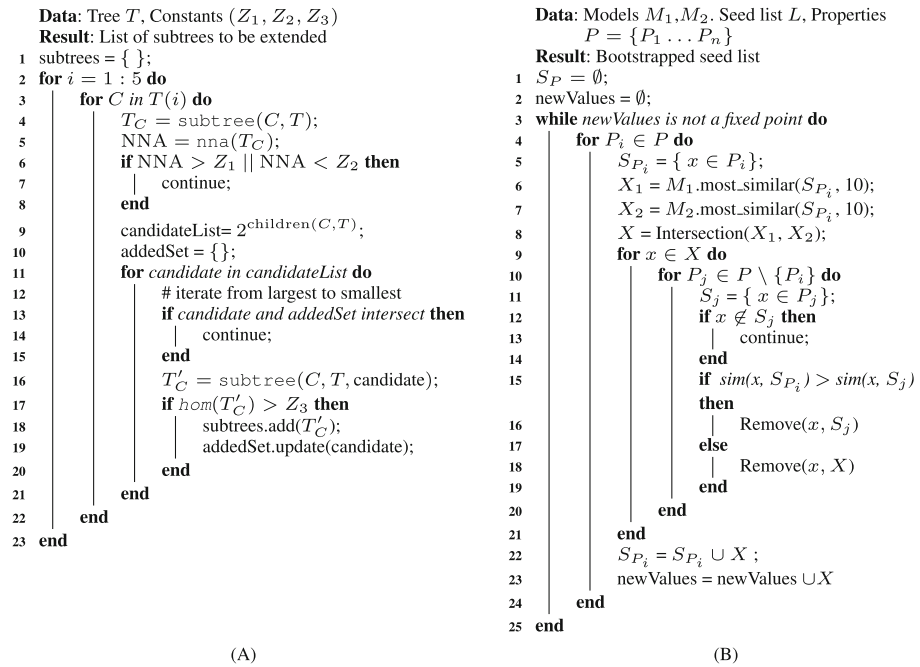


Fig. 5. Subtree selection and bootstrapping algorithms

immediately after the query. Then the subtree is said to be homogeneous if the number of such keywords is larger than $Z_3 = 30$, a value determined empirically.

In Fig. 5(A) we show the algorithm that we use to find the subtrees in the taxonomy. For the sake of clarity we use a number of functions that we will informally define next. Let T a tree, C a class in R, and M a model built as explained above, and S a set of classes. Then $T(i)$ returns the set of classes in T with depth i; `children`(C, T) returns the children of C in T; `subtree`(C, T) returns the subtree of T "hanging" from C; `subtree`(C, T, S) returns the subtree T_1 of T hanging from C but restricting the nodes with depth 1 in T_1 to those in S; `nna`(T) returns the effective number of items in T; `hom`(S) returns the homogeneity of T.

Property-Value Seed Extraction. We extract the initial set of properties and values (PV) from HTML tables and semi-structured text inputted by merchants in the items descriptions as these are easy to parse, quite accurate and reflect the domain knowledge of the merchants. There are several approaches in the literature to extract information from HTML tables [6,10,23], here we adopt the one used in [23] (Sect. 4.1.1) and use a slightly modified version of their implementation. Intuitively, the property names are first extracted from the headers of HTML tables in the item descriptions, and the associated values are found as the adjacent keywords either in the tables or in semi-structured text.

Model Training. The initial PV list obtained only from HTML tables and semi-structured text can lack a number of popular values, depending on the class. To increase the coverage of the property range we bootstrap the list using neural models of context similarity (word2vec). This module is in charge of training these models on the set of item descriptions within a given subtree.

We use word2vec's CBOW model with negative sampling to find words which appear in item descriptions within similar contexts. The item descriptions are first stripped from non-text features such as HTML tags and URLs. Then, they are tokenized using Mecab, a Japanese language morphological analyzer of which only the tokenizing part is used. Two models are trained on the resulting data. The first one is directly trained on the tokenized descriptions seen as bags of words. The second one is trained on the tokenized descriptions after performing collocation using popular search keywords extracted previously (by the Keyword Ranker). Collocation is done in two steps due to the specificities of the Japanese language. The first step is to join adjacent tokens into popular "words". Indeed, as the words are usually not separated by spaces or punctuation in Japanese, the tokenizer may cut words into several tokens. The second step is to join the resulting words into popular ngrams (up to trigrams).

After training we obtain two models trained on slightly different representations of the item descriptions. This module relies on the word2vec implementation of the library gensim.

5.2 Extension Layer

Seed Cleaning. The initial PV list extracted in the processing layer has usually a fairly high precision but is not usable as is. The first issue is the existence of redundant property names, meaning that the merchants use different words to identify the same concept. Redundant property names can either be *(i)* different terms, such as (manufacturer) and (maker), or *(ii)* the same term written in different alphabets or combinations thereof such as and (grape variety)[3]. Another issue is the existence of values that are not useful as discovery axes, such as expiration date, model numbers or long ingredient lists. It is critical to point out that this information might be accurate but is not deemed relevant for the purpose of this project. This is why we do not aim to obtain a complete model of the domain, but to extract the core fragment that is relevant to the users.

Properties Aggregation: We first remove redundant property names. For this we develop the following score function. Let P_1 and P_2 be two properties in the seed, m_1 and m_2 their respective range sizes and n the size of the intersection of their ranges. Their similarity score function is defined as:

$$L(P_1, P_2) = L_{\text{conf}}\left(\frac{n}{\min(m_1, m_2)}\right) \times L_{\text{size}}\left(\frac{\min(m_1, m_2)}{\max(m_1, m_2)}\right) - L_{\text{error}}\left(\frac{1}{n}\right)$$

[3] The Japanese language has three different alphabets: Hiragana, Katakana, and Kanji.

where L_{conf} is an increasing function representing the naive confidence that two properties are similar if they share many values respective to the maximum number of shared values, L_{size} is a decreasing function which tempers that confidence if the properties have comparable range sizes and L_{error} is an increasing function, with value 0 at 0, that increases as the number of shared values decrease, modelling the uncertainty over the score computed by L_{conf} and L_{size}. In practice we use $L_{\text{conf}}(x) = x$, $L_{\text{size}}(x) = \exp(-ax)$ and $L_{\text{error}}(x) = bx$, with the ad-hoc parameters $a = 0.33$ and $b = 0.1$. Two properties are considered similar if their similarity score is larger than 0.1 and the equivalence classes for this relation are computed. For each of these classes, we pick the representative that occurs more often in item descriptions as final property name.

Observe that [23,26] also performs an aggregation step. Intuitively, in [23] two properties are aggregated into a vector if they have a popular value (among merchants) in common. Two vectors are aggregated if the cosine similarity is above a given threshold. We empirically observed that the score function presented here can more accurately single out synonym properties since it does not depend on a single value occurring multiple times, but on the set of shared values and on the property sizes. In [26] they use cosine similarity the aggregate property names, again, in our experiments (using word2vec) we obtained better results using the score function presented here. As in [23], two properties are not aggregated if both are found to appear in the same item description during the seed extraction.

Values Filtering: The next step is to clean the properties' ranges by discarding any non-popular value, measured by their frequency in the search queries logs (obtained by the Keyword Ranker). The result of these two steps is a small list of property-values pairs with low redundancy and high precision which is representative of the interests of the users.

Bootstrapping. We then expand the coverage of the PV seed obtained so far. The bootstrapping algorithm, simplified for the sake of presentation, is shown in Fig. 5(B). We use two models, described in the previous subsection, to mitigate spurious high similarity between words that are not semantically similar as caused by the text pre-processing and tokenizing errors (particularly relevant for Japanese). Another use of the two similar models is to introduce a natural stopping condition to the bootstrapping algorithm. For each property, we only consider the 10 words most similar to the current range for each model then intersect the two outputs. This overcomes the problem of setting a meaningful threshold on similarities provided by word2vec.

The algorithm iterates over the property list, P, adding new values to the range of each property, S_{P_i}, until no more new values are found (newValues reaches a fixed point). For a new keyword x to be added to the property range S_{P_i} two conditions must hold: *(i)* all the models (two in this case) must agree that x is similar to S_{P_i} (lines 8–9); *(ii)* there should not be another property P_j such that x is more similar to S_{P_j} than to S_{P_i} (line 12). Observe that if x is

added to S_{P_i} and also belongs to a less similar S_{P_j}, then x is removed from S_{P_j} enforcing the disjointness of values in the property ranges.

Intuitively, the function most_similar finds the top n words in the vocabulary that are most similar to the words in the range of the property P. More specifically, it finds the top n vectors that maximizes the multiplicative combination of the cosine similarities (originally proposed in [15]) between the given set of vectors, S_{P_i}, and the candidate vector in the model vocabulary $V \setminus S_{P_i}$:

$$score_{M_i}^{S_P}(candidate) = \Pi_{v \in S_P} \texttt{cos(candidate,v)}$$

Triple Generator. This modules takes all the items titles/descriptions in a given subtree t_1, and the bootstrapped property-value list for t_1. For every item I and every property P, it first look for the values of P in the HTML tables and semi-structured text of the description of I, if it cannot find it looks for the values in the title of I, and finally it looks for the value in the free text in the description of I. If two different values for P appear together in one of the three steps above, it ignores them and moves to the next step. Once it finds a value v, for P in I, it generates the triple (I, P, v).

Semantic Gate. The semantic gate is in charge of exposing the triples through a SPARQL end-point, and moving the new extended ontology into OWL 2 (DL). Recall that the existing taxonomy is not available in any well-known ontology language. For the SPARQL end-point we use Sesame Workbench[4] and Ontop[5]. Ontop implements an ontology-based data access (OBDA) approach. Interested readers can look at [5,20]. We decided to use OBDA since it is a non-invasive way to introduce semantic standards (RDF/OWL/SPARQL) to the different business units in Rakuten, and it still allows the different departments to access the data through standard SQL, in which they are already proficient.

5.3 Limitations

The current implementation has two major limitations that we will work on in the future. The first one is that it only handles words as property values. Thus, alphanumeric values such as 100 ml or 2 kg cannot be handled at the moment, and therefore properties such as size are discarded. Extending our approach to handle this does not present any technical challenge but it requires time to implement it in such a way that it is not detrimental for performance. The second limitation of this implementation is that we only consider subtrees with a root with depth at most 3.

[4] http://rdf4j.org/sesame/2.8/docs/articles/workbench.docbook?view.
[5] http://ontop.inf.unibz.it/.

6 Experiments

At a high level, the system divides the taxonomy extension into the independent extension of subtrees. As such, we will use a two-fold approach to assess the system capabilities. First we look into the division of the taxonomy in relevant subtrees. Second we analyze the precision and coverage of the extension of the subtrees.

6.1 Subtree Selection

Our system selected 1,251 subtrees from the taxonomy. There is no way to quantitatively evaluate the correctness of the subtree extraction without putting the results in production and A/B testing to check the response of the users. Thus we will only briefly discuss the results of this process looking at the selected subtrees with a non-empty intersection with the subtrees hanging from either of these two dissimilar classes: wine and small fashion items.

In both cases the selected subtrees are found to hang from classes at depth 2, meaning that any class above that was too diverse. Our algorithm selects 4 subtrees containing classes related to wine. The largest subtree, denoted T_w below, includes red wine, white wine, sparkling wine, wine sets and "other wines". The other wine-related classes at depth 3 (rosé wine, non-alcoholic wine, wine accessories) are separated in the remaining 3 subtrees. The height of these subtrees varies between 2 and 3, therefore containing classes of depth up to 4 or 5 in the original tree. For classes related to small fashion items, all classes at depth 3 (neckties, belts, handkerchiefs, key cases, etc.) are separated in different subtrees. The subtree containing neckties is denoted T_n below.

In the case of the wine-related classes, the fact that non-alcoholic wines and wine accessories are separated from the others is consistent with the fact that they are semantically different. However, it could be argued that rosé should belong to the same subtree as red wine, etc. An explanation would be that shoppers searching for rosé wine are more knowledgeable of the domain and therefore use more specific terms in their searches, compared to more mainstream red, white or sparkling wines. Therefore, this separation allows to provide more specific properties to the shoppers browsing the rosé wine class. On the other hand, the classes related to small fashion items are very semantically diverse and would not benefit from being aggregated.

Subtree Extension. The extension of the taxonomy at the subtree-level consists of first extracting core properties and values (PV), then linking these to the items to generate triples. We evaluate these two steps separately.

Comparison to manual work: The PV extraction is currently done manually by domain experts, which we aim to supersede with our automated approach. A first quality criterion to compare both approaches is the recall, in this case the ability to discover all the relevant properties for a given subtree. The catalog team provided us with the outcome of the manual work on rice and beef, corresponding respectively to subtrees T_r and T_b below, which we used to summarize

Subtree	M \ A	M ∩ A	A \ M
T_r	1	3	3
T_b	2	3	6

(A)

Subtree	count	overall	max	median	mean	min
T_r	6	0.92	1.00	0.97	0.81	0.20
T_b	9	0.86	1.00	0.88	0.83	0.50
T_w	9	0.91	1.00	0.81	0.77	0.20
T_n	8	0.88	1.00	0.85	0.70	0.00

(B)

Fig. 6. Results for the property-value extraction

the overlaps of the extracted properties for both approaches in the Table (A) in Fig. 6.

The properties missed by the automated process (M\A) are: for rice, Size, which was discarded due to the alphanumeric values; for beef they are Brand, which was discarded due to the infrequency of the values, and Intended Use, which did not appear in the semi-structured text and is arguably not a property of the items. The automated process, on the other hand, found several properties that where missed by humans (A\M). In the case of rice, Shipping fee, Country of production and Composition. In the case of beef, Shipping fee, Ingredient, Product Name, Processing area, Country of production and Allergens.

For the properties which were extracted by both processes (M ∩ A), the total number of values for rice was 47 for the manual process and 102 for the automated process, while it was respectively 18 and 43 for beef. This, combined with the accuracy discussed below shows that the automated process increases the coverage by a large margin.

Accuracy of PV pairs: A second quality criterion for the PV extraction is the accuracy of the property-value pairs. Assuming naturally that the manual extraction by domain experts has a satisfying accuracy, we independently assess the accuracy of our automated process by manually annotating its results and checking a business minimal requirement of 80 %. In the Table (B) in Fig. 6 we analyze the accuracy in PV pairs extracted for the four subtrees T_w, T_n, T_r and T_b described previously. The table shows for each subtree: the number of properties, the overall accuracy of the pairs, and the distribution of the accuracies by property (max, median, mean, min). Observe that the overall accuracy is above 80 % in each subtree and the median property accuracies are also high. The properties with minimal accuracies in T_r, T_b, and T_w are properties with small initial ranges which were erroneously extended in the bootstrapping process. In T_n there is an ambiguous property that we could not evaluate its correctness and therefore assumed that all its values were wrong.

Analysis of triples: The triples are evaluated by annotating 50 randomly drawn triples for each subtree. Each triple was annotated with one of the three following labels: (i) "Correct" if the property and value are correct for the item; (ii) "Wrong pair" if the property and value would be incorrect for any item in the subtree; (iii) "Wrong linking" if the property and value would be correct for some other items in the subtree but not for the current item. We summarize the

proportions of these labels in the Table (A) in Fig. 7. As we can see, the accuracy is not consistently above our target of 80 %.

However, in the context of the business implementation of this work, a manual step will ultimately be added to identify the incorrect pairs in the PV extraction step (which is a limited amount of work, due to the moderate number of pairs), so that the final accuracy of the triples would be above 80 %.

Subtree	correct	wrong pair	wrong linking
T_r	0.74	0.08	0.18
T_b	0.68	0.16	0.16
T_w	0.86	0.12	0.02
T_n	0.78	0.10	0.12

(A)

Property	#values	eff. #values	coverage
Place of Origin	60	26.4	0.75
Country of Origin	15	2.3	0.24
Rice Variety	35	9.6	0.79
Composition	5	3.4	0.65
Category	7	3.9	0.57
Shipping Fee	4	2.5	0.53

(B)

Fig. 7. Results for the triples generation

We then compute the coverage for each property as the proportion of the items in the subtree with a triple containing this property. Another interesting measure is the effective number of values for a property. It is computed as $\exp(-\sum_v p_v \ln p_v)$ where the sum is over the values of the property (plus a catch-all "Unknown" value) and p_v is the proportion of the items linked to this property and the value v (for the "Unknown" value it is the proportion of items not covered by the property). In Fig. 7(B) we show these measures for rice (T_r) as well as the number of value per property. The coverage is found to be substantial with several properties over 50 %. Note how the property Country of Origin has 15 values but only an effective number of values of 2.3 as most of rice sold in Rakuten Ichiba is from Japan; the effective values being Japan and "Unknown" as most merchants do not feel the need to mention Japan, which is reflected in the low coverage of the property.

7 Conclusion

In this work we propose an end-to-end unsupervised approach that automatizes the process of extracting semantic information from text to: (i) extend the *relevant* fragments of a taxonomy with data properties deemed helpful to the user; and (ii) tag the items with this new information by generating the corresponding triples. We presented a novel technique to select relevant subtrees in the taxonomy in terms of overall profitability, ease of product discovery and similarity of users' shopping behavior, as well as a number of techniques to clean the text sources, aggregate equivalent properties and extend the range of the properties through text mining of semantic similarity. The approach presented

here can be easily ported to other scenarios beyond Rakuten Ichiba since it is language/data/technology independent. We provided a number of experiments showing the effectiveness of our approach in terms of precision and coverage.

In the future we plan to work on extracting object properties and axioms, lift the limitations discussed in Sect. 5, extract information from images, and trying to make publicly available an SPARQL endpoint with information about Rakuten Ichiba products.

References

1. Anantharangachar, R., Ramani, S., Rajagopalan, S.: Ontology guided information extraction from unstructured text. CoRR abs/1302.1335 (2013)
2. Barkschat, K.: Semantic information extraction on domain specific data sheets. In: Presutti, V., dAmato, C., Gandon, F., dAquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 864–873. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07443-6_60
3. Berardi, D., Calvanese, D., Giacomo, G.D.: Reasoning on UML class diagrams. Artif. Intell. **168**(1–2), 70–118 (2005)
4. Bing, L., Wong, T.L., Lam, W.: Unsupervised extraction of popular product attributes from e-commerce web sites by considering customer reviews. ACM Trans. Internet Technol. **16**(2), 12:1–12:17 (2016)
5. Calvanese, D., Cogrel, B., Ebri, S.K., Kontchakov, R., Lanti, D., M.R., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semant. Web J. (2016)
6. Chen, H.H., Tsai, S.C., Tsai, J.H.: Mining tables from large scale html texts. In: Proceedings of the 18th Conference on Computational Linguistics, pp. 166–172. Association for Computational Linguistics (2000)
7. Ding, Y., Fensel, D., Klein, M.C.A., Omelayenko, B., Schulten, E.: The role of ontologies in ecommerce. In: Handbook on Ontologies, pp. 593–616. Springer (2004)
8. Exner, P., Nugues, P.: Entity extraction: from unstructured text to dbpedia rdf triples. In: International the Semantic Web Conference, ISWC 2012, pp. 58–69. CEUR (2012)
9. Furche, T., Gottlob, G., Grasso, G., Guo, X., Orsi, G., Schallhart, C.: Real understanding of real estate forms. In: WIMS, p. 13 (2011)
10. Gatterbauer, W., Bohunsky, P., Herzog, M., Krüpl, B., Pollak, B.: Towards domain-independent information extraction from web tables. In: Proceedings of the 16th International Conference on World Wide Web, pp. 71–80. ACM (2007)
11. Giese, M., Soylu, A., Vega-Gorgojo, G., Waaler, A., Haase, P., Jiménez-Ruiz, E., Lanti, D., Rezk, M., Xiao, G., Özçep, Ö.L., Rosati, R.: Optique: zooming in on big data. IEEE Comput. **48**(3), 60–67 (2015)
12. Gupta, R., Halevy, A., Wang, X., Whang, S., Wu, F.: Biperpedia: an ontology forsearch applications. In: Proceedings of the 40th International Conference on Very Large Data Bases (PVLDB) (2014)
13. He, H., Meng, W., Lu, Y., Yu, C., Wu, Z.: Towards deeper understanding of the search interfaces of the deep web. World Wide Web **10**(2), 133–155 (2007)
14. Krestel, R., Witte, R., Bergler, S.: Predicate-Argument EXtractor (PAX). In: New Challenges for NLP Frameworks, pp. 51–54. ELRA (2010)

15. Levy, O., Goldberg, Y.: Linguistic regularities in sparse and explicit word representations. In: Conference on Computational Natural Language Learning, pp. 171–180. Ann Arbor, Michigan, June 2014
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
17. Mitchell, T.M., Cohen Jr., W.W., Hruschka, E.R., Talukdar, P.P., Betteridge, J., Carlson, A., Mishra, B.D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E.A., Ritter, A., Samadi, M., Settles, B., Wang, R.C., Wijaya, D.T., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J.: Never-ending learning. In: AAAI, Texas, USA, pp. 2302–2310 (2015)
18. Qiu, D., Barbosa, L., Dong, X.L., Shen, Y., Srivastava, D.: DEXTER: large-scale discovery and extraction of product specifications on the web. PVLDB **8**(13), 2194–2205 (2015)
19. Reinberger, M.L., Spyns, P.: Discovering knowledge in texts for the learning of dogma-inspired ontologies. In: ECAI 2004 Workshop on Ontology Learning and Population (2004)
20. Rodriguez-Muro, M., Rezk, M.: Efficient SPARQL-to-SQL with R2RML mappings. J. Web Sem. **33**, 141–169 (2015)
21. Saggion, H., Funk, A., Maynard, D., Bontcheva, K.: Ontology-based information extraction for business intelligence. In: Aberer, K., et al. (eds.) ASWC/ISWC - 2007. LNCS, vol. 4825, pp. 843–856. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_61
22. Schutz, A., Buitelaar, P.: *RelExt*: a tool for relation extraction from text in ontology extension. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 593–606. Springer, Heidelberg (2005). doi:10.1007/11574620_43
23. Shinzato, K., Sekine, S.: Unsupervised extraction of attributes and their values from product description. In: Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, pp. 1339–1347 (2013)
24. Wang, C., Kalyanpur, A., Fan, J., Boguraev, B., Gondek, D.: Relation extraction and scoring in deepqa. IBM J. Res. Dev. **56**(3), 9 (2012)
25. Wimalasuriya, D.C., Dou, D.: Ontology-based information extraction: an introduction and a survey of current approaches. J. Inf. Sci. **36**(3), 306–323 (2010)
26. Yoshida, M., Torisawa, K.: A method to integrate tables of the world wide web. In: International Workshop on Web Document Analysis (WDA 2001), pp. 31–34 (2001)