2019 IEEE 35th International Conference on Data Engineering (ICDE)

Accurate Product Attribute Extraction on the Field

Laura Alonso Alemany*
FaMAF
Universidad Nacional de Córdoba
Argentina
lauraalonsoalemany@unc.edu.ar

Lasguido Nio
RIT
Rakuten, Inc.
Japan
lasguido.nio@rakuten.com

Martin Rezk
RIT
Rakuten, Inc.
Japan
martin.rezk@rakuten.com

Ted Zhang
RIT
Rakuten, Inc.
Japan
ted.zhang@rakuten.com

***Abstract*—In this paper we present a bootstrapping approach for attribute value extraction that minimizes the need for human intervention.**

Our approach automatically extracts attribute names and values from semi-structured text, generates a small labelled dataset, and bootstraps it by extracting new values from unstructured text. It is domain/language-independent, relying only on existing semi-structured text to create the initial labeled dataset.

We assess the impact of different machine learning approaches to increase precision of the core approach without compromising coverage.

We perform an extensive evaluation using e-commerce product data across different categories in two languages and hundreds of thousands of product pages. We show that our approach provides high precision and good coverage. In addition, we study the impact of different methods that address specific sources of error. With error analysis we highlight how these methods complement each other, obtaining insights about the individual methods and the ensamble as a whole.

I. Introduction

Semi-supervised machine learning approaches have been extensively used in relation and attribute-value extraction [22], [10], [4], [24], [21], [17]. These approaches take a small or weakly annotated initial training set and iteratively extend this initial set with automatically tagged examples (a.k.a. bootstrapping).

One of the dangers of semi-supervised approaches is that they may increase coverage at the cost of sacrificing precision. In this work, we present a semi-supervised approach for Product Attribute Extraction (PAE), and assess its performance over Rakuten e-commerce data. In our business, case precision has priority over coverage, therefor in each bootstrapping iteration our approach implements specific methods that target well-known syntactic and semantic errors individually.

This work integrates and enhances different existing approaches to minimize the human supervision required for good coverage and high-precision (90% precision on average) while doing so in a domain and language-independent way (except for the tokenizer and part-of-speech (PoS) tagger). In our approach, the only human knowledge involved are four veto rules to identify and remove syntactically malformed values.

Our approach works as follows. An initial seed of *<attribute, value>* pairs is automatically obtained from HTML tables, but these examples cover only a small fraction of the products and attributes. That is why we apply a semi-supervised learning approach to bootstrap this initial seed. In each bootstrapping iteration, the classifier (we evaluated CRF and LSTM) trained on labeled examples from the previous iteration will produce new candidates. To correct automatically tagged examples, we use heuristics to first trim the values according to non-semantic and domain independent features. Later, a semantic distance between known and new values will be measured, removing newly tagged candidates which are dissimilar to previously tagged ones. This reduces semantic drift [7], a well-known problem associated with bootstrapping.

We have performed an extensive quantitative evaluation with real product data. We deployed an evaluation approach that allows us to assess the impact of different techniques.

In addition to quantitative evaluation, we have analyzed different factors affecting the performance of the system that, to the best of our knowledge, have not been specifically addressed in previous work on PAE: Do different product attributes behave differently w.r.t. precision and recall? Can we build an architecture that builds upon best performing attributes, and apply a different process to least performing attributes? How do different languages affect performance? How do different categories affect performance? How do different ML approaches affect performance? We have also looked into attributes whose nature is more complex than the standard attributes studied in existing work such as brand, color, or country of origin. Complexity in this case refers to how entities tend to be composed of several tokens including symbols, numbers, and letters, and different sellers tend to write each entity in several different ways. These qualitative explorations show the most promising future developments to improve performance.

Our main contributions can be summarized as follows: A domain/language independent end-to-end product attribute-name and attribute-value extraction architecture; a method to enrich the initial seed of attribute-values based on generalizing *via* diversification, which increases the accuracy of the approach up to ten points; a method to eliminate errors based on syntactic and semantic features; an extensive evaluation on real e-commerce data assessing the performance of our approach across different languages and categories; and a qualitative error analysis posing new questions in the area and addressing further developments.

It is also worth noting that unlike most existing work,

*Authors listed in alphabetical order.

2375-026X/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDE.2019.00202

we focus on non-English languages, namely Japanese and German.

The remainder of the paper is organized as follows: In the following Section we describe the business case, followed by some background. In Section IV we briefly show how our proposal builds upon related approaches. In Section V we describe our architecture and then the evaluation setting in Section VI. In Section VII and VIII we analyze the results of our experiments. Section IX concludes the paper.

II. 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.

Rakuten Ichiba, the leading e-commerce website in Japan, 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 catalog (in a form of a taxonomy, there are only classes and subclass relations between them) with around 40,000 classes. In addition, Rakuten Group owns e-Commerce sites in Germany, France, Taiwan and US; and each of them owns their own catalog with different levels of completeness, size, and complexity.

Currently there are several internal projects to improve the quality of these taxonomies, aiming to enhance user experience, increase conversion rate[1], and by doing so increase the gross merchandise sales per class.

In this work we focus on the project that aims to extend taxonomy classes and items with new semantic information. The focus is on product attributes and values deemed helpful to the user for search and retrieval, or to provide a more useful view of the product range.

To find such attributes and values, we analyze the product pages, more precisely, the titles and descriptions specified by the merchants. This text is HTML code, consisting mostly of free-form text, semi-structured text (table-like free-form text) and structured text (tables). This data is therefore the primary source to extract item-specific information in this project. It is also used to automatically acquire domain-specific knowledge as it reflects the knowledge of the merchants.

Since Rakuten has different catalogs in companies in different countries with different languages, we aim to develop a language and domain-independent approach that can be ported effortlessly. Since catalogs are very large, we need a procedure that guarantees a satisfactory coverage of product attributes. But since catalogs are used by customers in the process of their purchases, accuracy must be guarantee. Therefore, we must find a balance between precision and coverage of the system, two concerns that are usually opposed in Information Extraction systems. Our business need is that coverage is satisfactory but, most importantly, that precision is maintained as high as possible, even at the cost of losing some coverage.

[1]Proportion of customers making a purchase within a given browsing session.

III. Background

In this section we briefly define some terms that will be used in the remainder of the paper.

An *attribute* is a binary relation between products and values that describes characteristics of products in a given category. Attribute values might be formed by a number of tokens. For instance, the value $100\%\ cotton$ is formed by three tokens.

In this paper we focus on data attributes, that is, we do not consider attributes that describe relations between products.

A homogeneous category, is a category such that all its sub-categories have the same set of attributes. For instance, a category C consisting of red, white and rose wines is homogeneous. However, if the category also contains wine accessories with attributes such as "length", then C is no longer homogeneous. In this paper we focus on homogeneous categories.

Definition 3.1: ***Problem Definition:*** Let C be a homogeneous category, and A the set of attributes in C. Given a set of products $p_1 \dots p_n$ in C and their corresponding descriptions $d_1 \dots d_n = D$, we need to *extract* (i) a set of attributes $a_1 \dots a_m \subseteq A$ from D, and (ii) triples of the form $<p_j, a_i, e>$ such that the description d_j states that p_j has the value e for attribute a_i.

Observe that this problem requires finding the attribute names (since A above is unknown to us), the values, and distinguishing when a value corresponds to the product. For instance the sentence *this product does not include an Apple phone.* should not lead to the triple *<cellphone, brand, Apple>* , while if we remove *not* then it should.

In this project we do not check the truthfulness of what is stated by the description. Neither are we building a complete model of the domain, like the work in [16], but to use the extracted semantic information to improve the user experience and quality of the catalogs.

IV. Related Work

In general, a bootstrap Information Extraction (IE) system is initialized with a small set of seeds. These are labeled examples usually obtained by human annotation or through lexicons, rules, and/or patterns that the system can use on unlabeled data [25], [6], [10], [17], [19]. Others, such as [1], use an annotated corpus. In contrast, in our approach we obtain them automatically from HTML tables as in [13], [24], [2], [5], [11], [4].

Some other bootstrapping approaches are more akin to ours, but are limited in their scope. Putthividhya et al. [19] extract a fixed set of attributes from titles.

The works presented in [13], [24], [2], [5], [11], [18] propose similar semi-supervised frameworks that extract a set of properties and values from HTML semi-structured data. However, they apply neither attribute generalization (c.f. Section V) nor the syntactic+semantic cleaning steps we perform in each iteration. This cleaning is particularly relevant in

modern corpora which are large and varied, causing many noisy new instances, patterns or lexica can be added to the training set through bootstrap iterations. This phenomenon (noise getting amplified in each iteration) is known as *semantic drift* [7]. To address that, some approaches score candidate patterns [17], [12] by using distributional semantics, tf-idf or string edit distances for example. Other approaches apply error-correcting strategies to avoid semantic drift. Carlson et al. [3] is similar to our approach in that they use classifiers to label unlabeled data instead of using pattern extractors, and they impose constraints stating what the values should be for a given attribute to filter out new candidate entities. Our work differs from them in that our veto functions are not domain specific, and state what the values should *not* be. Qiu et al. [20] uses supervised learning to classify HTML fragments, and then combines two techniques to extract the property-values: the first strategy is similar to that in [24], and the second one is based on the outcome of various noisy annotators that rely on manually labeled data. More recent works are Ceres [15], which focuses on relation extraction from semi-structured data with distant supervision. OpenTag [26] uses LSTM and CRF to increase the attribute coverage by finding missing values for a given set of product attributes. In this work we discover both, attribute names and attribute values.

V. Overview of the solution

In this section we present our approach to PAE, and how we dealt with issues that arise in the e-commerce domain.

The system takes as input a set of product web pages, the users' search logs, and a set of domain independent functions to clean the data. As shown in Figure 2, these inputs are *preprocessed*, creating an initial labeled dataset. This dataset is fed to the *Tagger module*, which uses this data to train a ML model, which is used to identify new <*product_id, attribute, value*> triples in product web pages. Then, the *Cleaning module* revises these new triples and removes those that are flagged as errors by the cleaning functions or by the semantic similarity filter. Then, the remaining triples are added to the initial labeled dataset and fed to the Tagger. This Tagger–Cleaner cycle is iterated until a stopping criterion is met. In our experiments in Section VI, the stopping criterion was set to five iterations of the cycle. The algorithm of the process is detailed in Figure 1.

In what follows we describe each of the modules in detail.

A. Pre-Processor

This component (covered by lines 1-4 in Figure 1) produces an initial seed of attribute-value pairs from web pages. First, attribute name candidates are found in product pages, extracting attribute names and attribute values from HTML semi-structured data, mostly tables with a dictionary structure, that is, of 2 rows and n columns or of 2 columns and n rows, following the approach in [13], [24], [2], [5], [11], [4]. The outcome of this module is an (initial) candidate set of attribute-value pairs, with two main problems: redundant attribute names and incorrect attribute values.

Data : query logs ql, product pages W, veto functions, number of iterations N
Result: Triples

```
1  newValues = ∅;
2  candidates = candidate_discovery(W);
3  clean_candidates =
     value_cleaning(attribute_aggregation(candidates, ql));
4  complete_cc = value_diversification(clean_candidates,
     candidates);
5  dataset = training_set_generation(complete_cc);
6  attributes = attributes in complete_cc;
7  iteration = 0;
8  while iteration < N do
9      # tagging;
10     train_ds, tag_ds = split_dataset(dataset);
11     model = train(train_ds);
12     tagged_ds = model.tag(tag_ds);
13     # cleaning;
14     cleaner_ds = apply_veto(tagged_ds);
15     model_sem = train_semantic_model(cleaner_ds);
16     for a_i ∈ attributes do
17         core_{a_i} = find_semantic_core(a_i, cleaner_ds,
             model_sem);
18         clean_ds_{a_i} = remove_semantic_distant_values(a_i,
             cleaner_ds, core_{a_i});
19     end
20     dataset = clean_ds;
21     iteration = iteration + 1 ;
22 end
```

Fig. 1. Algorithm of the approach.

Redundant attribute names occur when several merchants use different terms to identify the same attribute. For instance, 製造元 (manufacturer) and メーカー (maker), or *black* and *schwarz* ("black" in German). [2] We aggregate similar attributes into a single attribute using the scoring function described in [4]. Intuitively, that function scores pairs of attributes according to the naïve confidence that two attributes are similar if they share many values respective to their maximum number of values, adjusted by a decreasing function which reduces that confidence if the attributes have comparable range sizes.

After that, incorrect attribute values are removed by keeping only those values that are found in search queries (from the search log input) or occur very often in its web page.

To improve the coverage of the initial seed, we developed a **value diversification** module (line 4 in Figure 1), one of the contributions of this work. This module tries to diversify the shape of values, for instance: if there are only integer values for a given attribute, the system will look for decimal values, or ranges of values. To do that, the system takes the most k frequent POS-tag sequences for each attribute

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

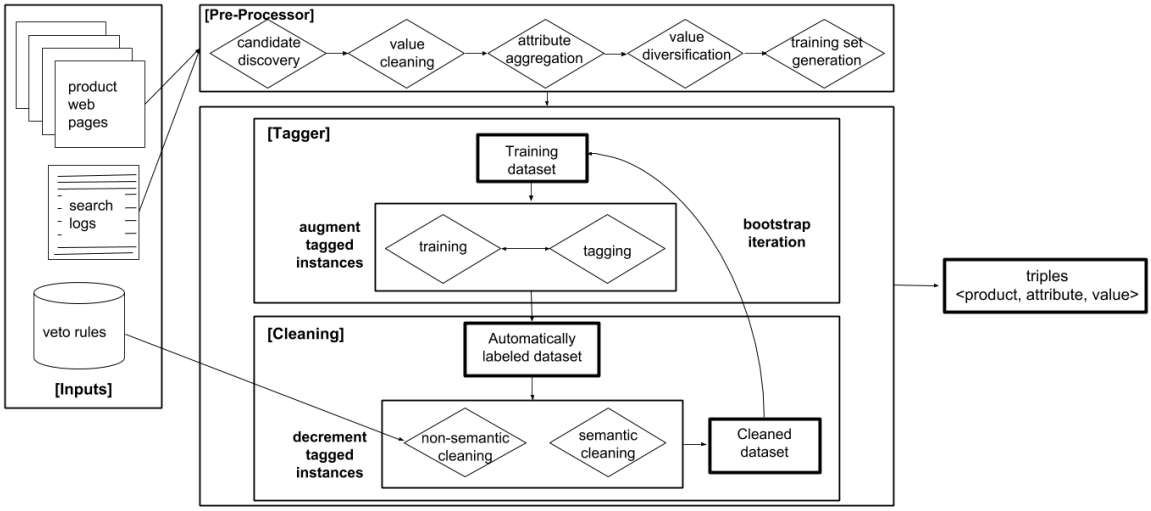

Fig. 2. Diagram of the flow of our approach.

(e.g. number-symbol-number-NN for 1.5kg)[3], and for each sequence we take the n most frequent values for that sequence in that attribute. From the output of this module we get a concise and clean set of tuples that provides an initial abstract representation of the category. We show the impact of this module in the next section.

The training set generation (line 5 in our algorithm) takes the seed and tags an initial set of products (the few ones with dictionary tables) with it. When we have our set of triples <*product_id, attribute, value*> , we label product web pages by tokenizing all the sentences in the product title and descriptions of $product$ and tagging all occurrences of $value$ with $attribute$, where $value$ may be a multiword.

B. Tagger

This component (covered by lines 10-12 in our algorithm) tags new triples in the corpus of product webpages by training a machine learning model from a labeled dataset which is either the seed dataset or the tagged dataset from a previous bootstrap iteration.

C. Cleaning

This component (covered by lines 14-20 in our algorithm) eliminates possible errors by removing tags that have been automatically added to the dataset and that are found unreliable. The early removal of probable errors prevents a snowball effect that leads wrongly tagged items to proliferate in future iterations. It consists of two modules:

• The *non-semantic cleaning* module discards tags using the set of *veto functions* provided by the user. In our setting we used the following veto functions: *(i)* symbols: 1-gram entities that are symbols such as ";" or "*". *(ii)* mark-up tags. *(iii)* unpopular entities: Per each attribute, we order the entities by the number of items that have been tagged with that entity, and keep only the top 80%, similar to [23]. *(iv)* long values: values that exceeds 30 characters.

• The *semantic cleaning* module prevents semantic drift over bootstrapping iterations. Concretely, a new value with tag a should be semantically similar to other values that are tagged as a. To measure semantic similarity between words, we use word2vec trained with our own corpus of product web pages in *each iteration*. Since these values/entities are domain-language specific, we can neither re-use existing word-embedding models (for example using Wikipedia data) nor can we use models trained from previous iterations of the loop, since they would not account for newly discovered entities.

We do this in three steps: *(i)* Group multiword attribute values tagged by the model as a single word. *(ii)* Create a core set of entities for each attribute. *(iii)* For each attribute, we take the vector representation of the values for the attribute and iteratively discard the value with lowest cosine similarity with the rest of values, until only n values are kept.[4]

VI. Evaluation setting

In this section we evaluate our approach and the impact of different modules on the overall performance of the system.

A. Datasets

Evaluation is done per category, since each category has its own set of attribute-value pairs. We evaluated 21 different categories in Japanese and German, 18 for Japanese and 3 for German, totaling 200k product pages. For each selected category in Japanese we obtained a dataset with 10K items in average, ranging from 4k to 12k. German datasets were around 2K in average. In addition, we studied the impact of different levels of homogeneity of the products in two pairs of subcategories of these categories.

[3]Japanese Pos tagger splits 1.5 into three tokens.

[4]We take the multiplicative combination of the cosine similarities of all the elements in the core set $\cup\{value\}$

Although the datasets used in this evaluation are not publicly available, Rakuten releases product datasets that can be found online[5].

B. Ground Truth

It is important to note that evaluation of Information Extraction systems is costly and often inaccurate. Creating a ground truth evaluation dataset requires intensive manual annotation with human experts. Therefore, building a representative test dataset is beyond the possibilities of most projects. Only publicly funded challenges offer a methodologically thorough opportunity for evaluation. In an industrial environment, a compromise between costs and comprehensiveness of evaluation needs to be met.

In our project, we reduced the cost of building an evaluation corpus by generating a truth sample as follows: Using an early version of the system presented in this paper, we automatically obtained large sets of *<product,attribute,value>* triples from categories that are relevant for the business. These triples were given to human annotators to verify correctness. For each triple, annotators stated whether a given *<attribute,value>* pair was a valid association (e.g. *<color,pink>*) and whether the triple *<product,attribute,value>* was correct (e.g. *<handbag_287,color,pink>*). For each category, $2k \sim 20k$ triples were evaluated for Japanese, and $500 \sim 1000$ for German. The total size of the truth sample is 235,825 triples.

The truth sample produced with this method has the advantage of providing a picture of the performance of the system across a wide variety of categories, attributes and values. This is an important difference with most IE and particularly PAE efforts, where evaluation is anecdotical and restricted to simple attributes such as brand or color, without much variation in the values and without complex linguistic phenomena. In contrast, we are evaluating with a large number of triples, covering heterogeneous domains and with complex, multi-word values for attributes.

The main disadvantage of this dataset is that it is biased with respect to the recall of the system. Since the triples to be manually validated have been obtained with the system itself, it is difficult to evaluate how many attributes are left out, therefore it is difficult and to evaluate the actual recall of the system. Creating a dataset where recall can be measured is much more costly. Observe, however, that this evaluation method serves well our most important objective: high accuracy. Even if we are trying to increase the coverage of *<attribute, value>* pairs associated to products, we need to keep precision high. Thus we need to measure precision, and coverage of the products. Recall is not evaluated without bias because it is too expensive and not the highest priority.

Thus, whenever we are employing the term "coverage" in the remainder of this work, we are referring to coverage of the triples identified by an earlier version of this system that were labelled as correct, and not to actual recall.

[5]https://rit.rakuten.co.jp/data_release/

C. Metrics

For the business need of Rakuten, it is very important that the output of the system is as close as possible to error-free. Therefore, the evaluation of different modules is targeted to find the parameters that minimize the number of errors. Thus, the main metric to evaluate the system is precision, that is, the proportion of correct *<product, attribute, value>* triples over the total number of triples produced by the system.

> *correct = triples generated by the system that also occur in the truth sample marked as correct.*
> *incorrect = triples generated by the system that also occur in the truth sample marked as incorrect.*
> *maybe_incorrect = triples generated by the system, such that the product id and the attribute name coincide with some correct triple in the truth sample, but it disagrees in the value.*

To illustrate *maybe_incorrect* triples , suppose that the system generates *<handbag_287, color, pink>* , but in the truth sample we have *<handbag_287, color, white>* . This new triple might actually be correct as well, however, we assume it is wrong.

$$\text{precision} = \frac{\#\ correct}{\#\ correct + incorrect + maybe\_incorrect}$$

We have noted that the first iteration of the bootstrapping cycle, that is, the first automatically tagged and cleaned dataset that is produced, has the biggest impact on the overall precision of the system. That is why very often we report on the precision after that first iteration only. This allows to obtain an early diagnostic to find successful configurations of the system.

As an alternative measure to recall, we obtain the coverage that a given attribute has over the universe of products. This metric, similar to the *support* metric for Association Rules, is a useful measure for business. It is calculated as the proportion of products in the category for which a triple has been found. From a business perspective, it means how many products can be found by faceted search using the newly discovered semantic information.

$$\text{coverage} = \frac{\#\ \text{products with a triple}}{\#\ \text{products in the input dataset}}$$

Note that if a product is covered, it does not mean that all its attributes are tagged or that they are tagged correctly. Observe that only a fraction of the products intersect with the truth sample, therefore we cannot accurately measure how many products are covered only by correct triples, but with the precision and the coverage this number can be approximated when needed.

D. Machine learning approaches

Bootstrapping approaches are in general based on machine learning methods. We instantiated our system with two different machine learning methods: Conditional Random Fields (CRF) and Recurrent Neural Networks (RNN).

CRFs [14] are a type of discriminative undirected probabilistic graphical model. In NLP it is often used to labels words in a sentence, but in principle it can model any sequential data. In particular we use CRF with limited-memory BFGS training algorithm with L1+L2 regularization, the default configuration. Since we aim to create a language/domain independent approach, we use general and standard[6] features to create the vectors for CRF, that is, for a given token/word in position t (w[t]) we generate the following features: the word w[t], the words in a windows of size K around w[t], the part-of-speech (pos) tags of such words, the concatenation of the pos of those words, and the sentence number.

Recurrent neural network [9]: are a type of discriminative network model where connections between units form a complex directed graph along a sequence. Specifically, we utilize Bidirectional Long Short-Term Memory (BiLSTM) type of RNN. For this evaluation, we use the implementation provided by NeuroNER [8]. NeuroNER stacks 2 kinds of LSTM in the hidden layer to compute both previous and forward context of sequence input. It uses Stochastic Gradient Descent (SGD) with dropout regularization to update the weights. During the experiment, we use both character and word level embedding. Character level representation is used as an input to BiLSTM, and word level representation is appended to the BiLSTM output to enhance the embedding layer. Later we feed this embedding layer to feed-forward network, resulting in a list of label probability.

Basically, we used both systems out of the box to minimize human intervention and to emphasize the portability of the approach.

We set the stopping criterion for the bootstrapping iterations at a maximal number of 5 iterations.

VII. Quantitative Analysis of Results

In this section we provide a set of results showing the precision and the coverage of our approach across the different categories. First, in Section VII-A we will study the seed and the initial training set, then the results of each iteration of the system in Section VII-B. In Section VII-C we will look into the number of triples generated by the system. In Section VII-D we will show a detailed account of the impact of various modules in the performance of the system, and finish the section summarizing the results.

A. Precision and Coverage of the seed

As explained in Section V-A, we automatically obtain the tagged instances to train the initial model for the first bootstrapping cycle. For each category, tuples of the form *<attribute, value>* (set S_1 above) are obtained by parsing HTML tables that have dictionary form, and with that we tag sentences in the text of the product pages that have that value or attribute. The size of the seed is around 2500 pairs of triples per category in average, ranging from 500 to 5000, and covering from 1% of the products in Garden, to almost 40% of the products in Ladies bags.

Since this is a fully automatic procedure, it is necessary to assess its precision, as the quality of the seed severely impacts the whole performance of the system. This is because the errors in the seed produce a snowball effect, and propagate across the iterations. Recall that a given *<attribute,value>* pair is correct if it is a valid association.

In Table I we can see that the precision of the initial seed is already high, averaging 95%. Unsurprisingly, the categories where the seed is more error-prone, like Garden, obtain worse results in the bootstrapping iterations, as can be seen in the following sections. Note that we make a difference between pairs *<attribute, value>* and triples *<product, attribute, value>* . The precision for triples including the product is lower than for pairs, as can be expected.

It can also be seen that the coverage of the obtained triples (with respect to what was covered by a first version of the system, and labelled as correct by human evaluators) is very small. That's why a bootstrapping procedure such as the one presented in this paper makes a big impact in the performance of the system.

B. Precision and Coverage of Bootstrapping

In Figure 3 we can see the precision and coverage across iterations of the bootstrap cycle of CRF, with and without cleaning. We can see that precision decreases a little with bootstrap iterations, but cleaning keeps precision above 85% in most cases, and it remains at the same rate for categories that start with a very high precision in the seed and first iteration, like Tennis, Ladies Bags or Digital Cameras.

We can also see that coverage is highly increased across iterations, although this increase is lower with cleaning.

In Tables II and III we provide more detail on the performance of the system across 8 categories after the **first** iteration of the bootstrap cycle. As we have seen above, the quality of the results obtained after the first cycle of bootstrap is the most indicative of the quality of the overall solution, after the quality of the seed.

In Table II we can see that, in general, precision figures are very good, around or over 90%. CRF tends to obtain better results than RNN, but with the cleaning modules they obtain comparable results. However, RNN has a tendency to overfit, which can be seen in a decrease of precision when trained with 10 epochs instead of 2. This impacts more severely categories with low precision as opposed to categories with high precision, such as Digital Cameras.

Most importantly, we can see that the cleaning module systematically makes an improvement in precision. In categories where the precision of the machine learning method is already high, as is the case of Digital Cameras, the cleaning does not have much room for improvement, but it does not degrade the performance.

Concerning coverage, in Table III we can see that the same configuration of RNN that produces very low precision has very high coverage. Since the target of the system is precisely

[6]See for instance www.chokkan.org/software/crfsuite/tutorial.html

	Tennis	Kitchen	Cosmetics	Garden	Shoes	Ladies bags	Digital Cameras	Vacuum Cleaner
#Pairs	296	467	613	196	156	723	224	509
#Triples	2109	1394	6655	952	697	5156	2157	2135
Precision Pairs	100	94.06	100	92.08	93.02	98.45	95.55	94.96
Precision Triples	98.76	93.03	93.08	88.52	92.09	98.05	99.74	96.45
Coverage Triples	25.50	19.50	36.61	8.3	6.47	39.15	12.14	27.25

TABLE I
PRECISION AND COVERAGE FIGURES FOR THE AUTOMATICALLY OBTAINED SEED INSTANCES TO TRAIN THE MACHINE LEARNING MODEL IN THE FIRST BOOTSTRAPPING ITERATION.

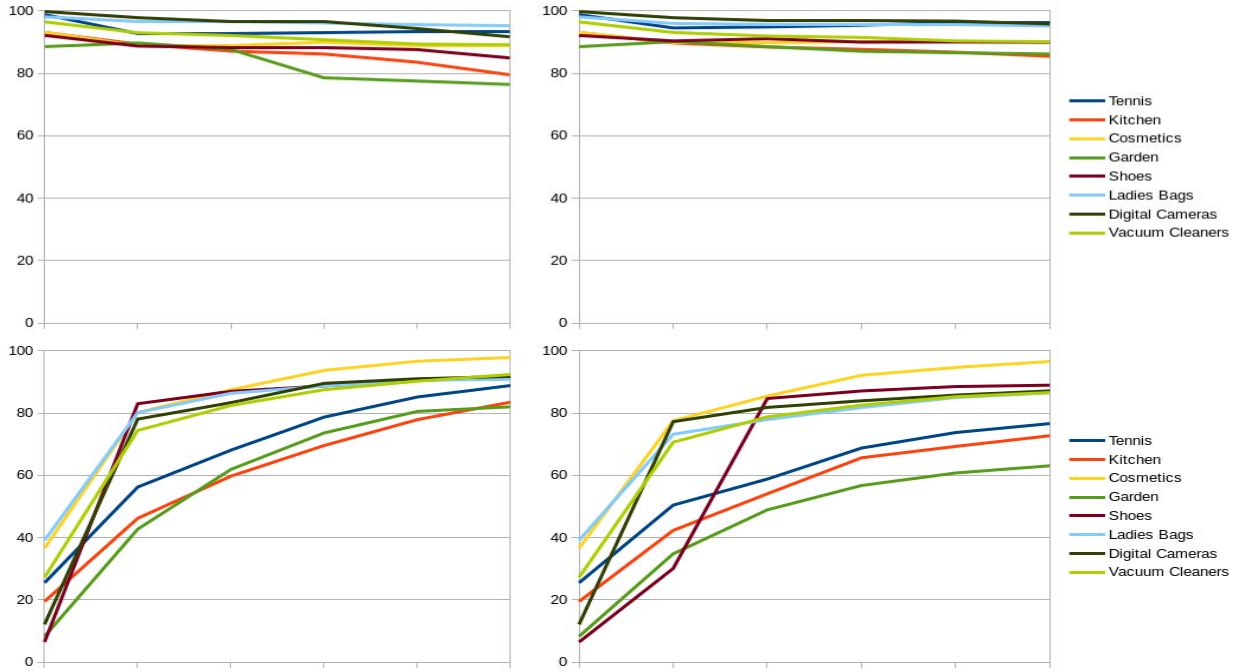

Fig. 3. Precision (top) and coverage (bottom) of the CRF model across bootstrap iterations, without cleaning (left) and with cleaning (right).

precision, this approach is discarded. If we compare the two tables, we can see that precision is inversely correlated with coverage. Indeed, for cases with extremely high precision, like Digital Cameras with RNN with cleaning (99.9% precision), we find only 16.59% coverage, while for the 40.33% precision obtained by RNN with 10 epochs for Cosmetics the coverage is 99.65%. Since our business requirements specify high precision, we prioritize approaches with higher precision and decent coverage, which in this case is CRF with a precision of +95% and a coverage of 87% after 5 iterations.

In the case of German we obtained similar results for the categories we inspected: *mailbox*, with a precision of 94.36% and a coverage of 73%, *coffee machines* with a precision of 92% and a coverage of 57.3% and *garden* with a precision of 84.2% and a coverage of 87.03%.

C. Number of triples generated

In this section we will look into the actual number of triples generated by the system, a feature related to the coverage of the system.

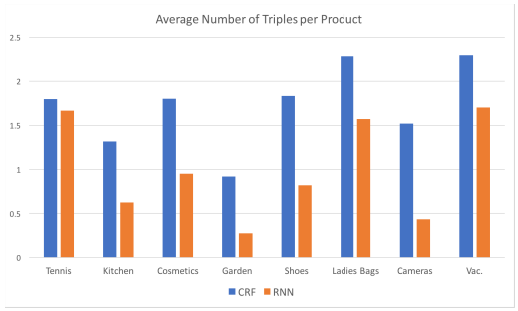

Fig. 4. Average number of triples per product obtained by the two different ML approaches after the first bootstrap iteration, including cleaning.

In Figure 4 we can see the average number of triples per product obtained by the two different Machine Learning approaches (CRF and RNN) after the first bootstrap iteration, including cleaning. We can see that CRF consistently associates more triples to products.

In Figure 5 we show the the total number of triples per

-	Tennis	Kitchen	Cosmetics	Garden	Shoes	Ladies Bags	Digital Camera	Vacuum Cleaner
RNN 2 epochs	81.29	83.61	91.66	64.22	83.45	85.09	99.45	80.28
RNN 10 epochs	40.29	77.04	40.33	76.62	53.92	76.12	98.36	74.80
RNN 2 epochs + cleaning	89.77	88.06	**91.61**	75.53	**91.22**	**96.25**	**99.94**	87.46
CRF	92.75	89.30	88.97	89.69	88.69	96.56	97.79	92.96
CRF + cleaning	**94.51**	**89.71**	89.81	**90.14**	90.36	95.97	97.79	**93.05**

TABLE II
PRECISION FOR THE FIRST BOOTSTRAP ITERATION OF VARIOUS CONFIGURATIONS OF THE SYSTEM TO OBTAINING ATTRIBUTE-VALUE PAIRS ACROSS DIFFERENT CATEGORIES: USING CONDITIONAL RANDOM FIELDS (CRF) OR RECURRENT NEURAL NETWORKS (RNN) AFTER A DIFFERENT NUMBER OF EPOCHS, WITH OR WITHOUT CLEANING OF THE OBTAINED DATASET.

-	Tennis	Kitchen	Cosmetics	Garden	Shoes	Ladies Bags	Digital Cameras	Vacuum Cleaner
RNN 2 epochs	85.85	57.8	85.86	39.9	54.17	90.67	16.92	88.4
RNN 10 epochs	99.65	75.31	99.65	45.11	83.28	91.44	22.29	95.31
RNN 2 epochs + cleaning	79.37	46.96	80.14	23.84	47.26	80.95	16.59	73.2
CRF	56.26	46.21	80.18	42.73	83.01	80.14	78.07	74.43
CRF1 + cleaning	50.45	42.32	77.53	34.82	30.11	73.2	77.24	70.65

TABLE III
COVERAGE OF THE UNIVERSE OF PRODUCTS IN EACH CATEGORY, FOR THE FIRST BOOTSTRAP ITERATION USING CRF OR RNN, WITH OR WITHOUT CLEANING OF THE OBTAINED DATASET.

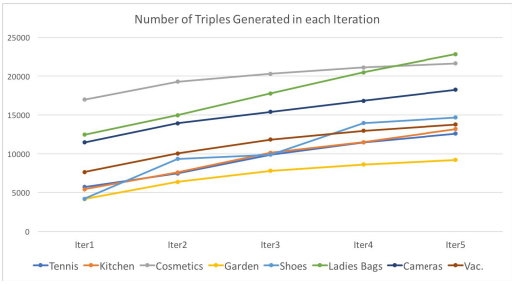

Fig. 5. Number of triples of different categories through bootstrap iterations, using CRF with cleaning.

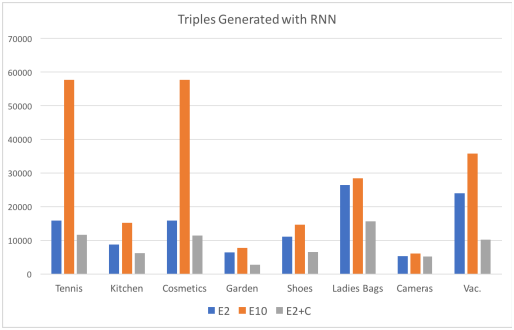

Fig. 6. Increase in the number of triples after the first iteration of the bootstrap cycle of different configurations of RNN: RNN with 2 epochs, 10 epochs and 2 epochs with cleaning.

category through bootstrap iterations, for the CRF model. We can see that there is a steady increase that would yield decreasing gains should the iterations continue.

In Figure 6 we show the results for the increase in the number of triples after the first bootstrap cycle for different configurations of RNN: with two epochs, with 10 epochs and with two epochs with cleaning. We can see that the approach with 10 epochs increases the number of triples much more than the other two, but, as shown in Table II, it does so at the cost of significantly reducing the precision of the system. The approach with 2 epochs and cleaning systematically has a smaller increase, but maintains high precision. As stated before, although we want to increase coverage, we cannot afford to sacrifice precision.

We can also see that in average both approaches find less than three properties per product. We will analyze this limitation and outline a strategy to address it in Section VIII-D.

In the case of German we obtained 2096 triples for garden 2943 triples for mailbox and 1626 triples for coffee machines.

D. Impact of different modules in the performance of the system

We also want to assess the impact of different components of the system. In Table IV we display a detailed evaluation of different configurations of the system for 2 very different categories: Vacuum Cleaner, a category where product attributes are rather well specified, and Garden, a category where descriptions are less rich.

The results in the top half correspond to the first iteration of the bootstrap cycle. The precision is high for both of them, around 90% if CRF is used. RNN obtains precision of around 75%.

As hinted before, we can see that the modules for syntactic and semantic cleaning and for value diversification make an important contribution to improve the performance of the system. This impact is much bigger for Garden than for Vacuum Cleaner, because the precision for Vacuum Cleaner leaves little room for improvement.

Garden starts with less than half the seed instances than Vacuum Cleaner, and with a much smaller coverage (8% coverage for Garden vs 27% coverage for Vacuum Cleaner). Thus it is much more difficult to obtain good precision for

After the first bootstrap cycle

	Vacuum Cleaner	Garden
RNN	74.8	75.53
CRF full	93.1	90.14
CRF-sem	92.94	83.33
CRF-sem-synt	91.87	80.33
CRF-div	91.18	87.90

After the fifth bootstrap cycle

	Vacuum Cleaner	Garden
CRF full	86.49	86.17
CRF-sem	87.93	76.4
CRF-sem-synt	76.92	67.69
CRF-div	75.74	85.98

TABLE IV
PRECISION OF DIFFERENT CONFIGURATIONS OF THE SYSTEM AFTER THE FIRST CYCLE OF BOOTSTRAP (TOP) OR AFTER THE SECOND CYCLE OF BOOTSTRAP (BOTTOM), REMOVING SOME OF THE MODULES THAT INCLUDE OR EXCLUDE EXAMPLES FOR TRAINING (SEMANTIC CLEANING "SEM", SYNTATIC CLEANING "SYNT", VALUE DIVERSIFICATION "DIV").

Garden. We observe that most of the gain is in performance in the application of semantic cleaning, without that module, there is a drop in precision of 6.5%. If syntactic cleaning is also removed, there is an additional loss of 3%. When we remove value diversification, the loss in performance is of only 2% of precision.

For Vacuum Cleaner, even if the differences in performance are not as noticeable, we can still see that removing any module produces a decrease in performance, thus the modules have a positive impact.

The results in the bottom half correspond to the fifth iteration of the bootstrap cycle. These have been obtained by applying all modules up to the fourth cycle (except for diversification that is applied one during seed creation) and assessing the impact of each module only in the fifth cycle. We can see that, even if all modules are used, the precision has decreased to around 86% for both categories. We can still observe differences between Vacuum Cleaner and Garden. For Garden, the same tendencies as in the first iteration hold: the module with the biggest impact is still semantic cleaning, whose removal causes a drop of 10% in precision, but removing syntactic cleaning as well makes yet an additional 10% drop in precision. The value diversification module does not impact Garden, but it has a 10% impact in Vacuum Cleaner, as well as syntactic cleaning. In contrast, removing the semantic cleaning module produces an improvement of 1.5% in Vacuum, probably due to the fact that this category has very homogeneous values for its attributes and semantic cleaning is useful to capture and restrict more lexical variations.

To conclude, while the gain in performance for some modules is not as dramatic as for others, none decreased performance significantly, and their combination offers good results.

E. Summary

The overall precision and coverage (at the product level) of the system is high across categories and languages.

The system automatically finds a seed of training examples with good precision and coverage, which is a good starting point for the bootstrapping procedure. The precision falls slightly across iterations, and coverage is increased significantly.

Cleaning, especially semantic cleaning based on word similarity, has a big impact on maintaining high precision, while still allowing for increases in coverage. This impact is more evident in categories with noisier or smaller seeds, like Garden.

Regarding the machine learning approach, both CRF and RNN perform comparably given the right configuration of RNN. RNN approaches with many epochs tend to overfit, but we can obtain good results with only two epochs. However, using both system out of the box, CRF tends to be more stable and finds a good compromise between enhancing coverage and keeping precision high, which is preferable for business purposes.

RNN and especially the combination of both approaches have much potential especially to improve the property level coverage. We will investigate this in future work.

VIII. QUALITATIVE ANALYSIS OF RESULTS

As a general observation, we see that precision figures are often affected not by a large number of different errors, but a few errors that affect many items. This makes it easier to improve performance, whether it be by applying new heuristics or by manual intervention, like modifying the seed corpus or by correcting the output manually (human-in-the-loop).

A first source of errors is due to the fact that merchants may describe several products in the same page. When this happens, the system may find an *<attribute,value>* pair that is semantically correct, say *<color,red>* , but that pair does not refer to the main item being sold in the page, but to a secondary item recommended in the text.

A second source of errors are attributes with similar ranges of values, which the model tends to confuse. Examples of these are optical zoom vs digital zoom, or total pixels vs effective pixels (in Digital Cameras), length vs width (in Rings, not shown in this paper), etc. In the Garden category, the maximum weight allowed for shipment is confused with the weight of the product.

A. Impact of value diversification

To address the lack of coverage of the seed, we have implemented the value diversification module (see Section V-A). For instance, in the category Vacuum Cleaner, disabling this module drops down overall precision for CRF from 86% to 75% (c.f. Table IV). The main problem in this category occurs with the attribute 'weight'. For this attribute, integer numbers are much more popular in Vacuum Cleaner descriptions than decimal numbers, therefore no decimal number is sampled in the initial seed. This leads the CRF model to wrongly tag only the decimal part of the number as the attribute value together with the unit since it is the pattern given in the training set, basically integer+unit (5kg for the string 2.5kg). The RNN model on the other hand simply does not tag any value containing decimals. Thus, without diversification the coverage for that property drops from 40% to 1% . In particular, with the

value diversification module on, it finds 1068 different values, including decimals values. When the module is off, it finds 166 different values, all of them integers.

A similar situation happens with effective numbers of pixels in Digital Cameras. Sometimes the merchants use the thousands separator, but the ML algorithm fails to generalize and recognize the whole number as an entity. Thus, for $2,430$ it tags only 2[7].

B. The impact of cleaning

As explained in Section V-C, the elimination of probable errors consists of two parts: the non-semantic and the semantic cleaning.

The non-semantic cleaning consists of very simple veto rules that discard strings that cannot possibly be a value, such as symbols, mark-up tags or unpopular values for the attribute. They tend to discard 10% of the candidate triples in the first iteration, more or less the same amount of triples discarded by the semantic drift heuristic. For example, for the category of Digital Cameras, with over 5500 triples in the human-created ground truth, the first iteration of bootstrapping obtained 14156 candidate triples, of which 10% were removed by veto rules.

In the quantitative results we have seen that semantic cleaning increases the precision of the system. It effectively removes values that are not part of the range for an attribute.

For instance, in the Garden category, without semantic cleaning, the tagger often confuses colors, for example "yellow", with the term "flower shape" (花形). With semantic cleaning, "flower shape" is removed as part of the possible values for the attribute "color", which reduces this kind of errors.

We did parameter exploration for semantic cleaning, more concretely, we modified the parameter n that selects only n elements to determine the semantic core of values for an attribute (c.f. line 20 in Figure 1). This varies the strictness to calculate the semantic core of an attribute: if n is very small, we will be discarding many values that are not similar to a small core of very similar values for that attribute. If n is very big it might allow semantic drift.

However, we found that having no restriction on n did not heavily reduce the precision of the system 1% in the worst cases (Garden and Shoes). This seems to indicate that most of the new values that the system produce are semantically close to each other. This is because of the strict way in which the system obtains the values.

C. Precision per attribute

In this section we will study attributes that challenge the system more than the standard brand, color, and provenance (made in). Recall that these values are input by the merchants.

Within the category of Digital Cameras we studied (A_1) shutter speed, (A_2) number of effective pixels, and (A_3) weight.[8]

These attributes are challenging in that they do not occur in every camera description. For shutter speed the format of the values varies a lot, e.g. 1/6000秒～30秒 , 1/4000, 1～1／1600秒 , etc. The effective pixels attribute is very similar to the total pixel attribute, which can be misleading for the system, and the weight tends to be confused with other weights in the page as explained before. Nevertheless, we obtain a precision of 100% for shutter speed, 90% for the number of effective pixels and 100% for weight.

Similar situation happens in Vacuum Cleaner. In this category we studied (B_1) type, (B_2) type of container, and (B_3) and power supply type.[9] Again we have high precision for the three attributes, over 90% for type and type of container and 87% for power supply type.

However, the coverage of these attributes is rather small, around 10% in average. In the next section we tackle this issue.

D. Coverage per attribute

It is worth noticing, that not every product description contains attribute information. Therefore, if a product is not tagged with a particular attribute, it can be that the attribute is just not described in the product page. The fact that we cannot assess recall is a limitation of this evaluation.

So far, the results we presented were obtained by a single *global* CRF/RNN model that tags all the different attributes of the category. To increase the coverage we trained *specialized* models that tags only a subset of these attributes. In Figure 7 we show the increase in coverage of these attributes for the camera category when tagged using a single general model (+g for global in the attribute name) or when using one specialized model for a subset of them (+s for specific in the attribute name), namely, A_1, A_2, and A_3 above. In Figure 8 we look into the Vacuum Cleaner attributes.

The specialized models can increase the attribute coverage by, in some cases, orders of magnitude.

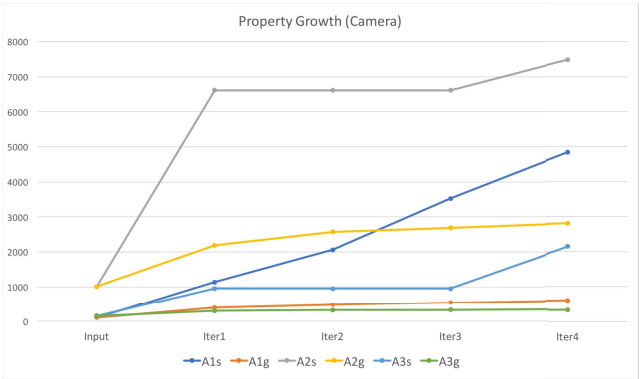

Fig. 7. Increase in the coverage of different attributes - Camera

In light of this, we evaluated the performance of the system when we obtained separate models for each attribute. Separate

[7]This a simplified English translation, original correct value is 約2,420万画素 .

[8]In Japanese シャッタースピード , 有効画素数 , and 重量 respectively.

[9]In Japanese タイプ , 集じん方式 , and 電源方式 respectively.

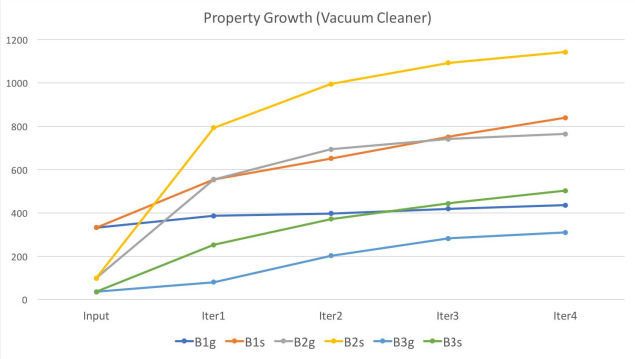

Fig. 8. Increase in the coverage of different attributes - Vacuum Cleaner

models, however, reduced global precision. For instance, for the attribute B_3, power supply type, in Vacuum Cleaner, the precision goes from over 90% to below 70%. Most of the errors come from confusing the attribute type with this one. Observe that the values for these two attributes are considerably different in general. This confusion does not occur when training the model with both properties together.

This suggest that the ML model uses the distinction between attributes to better tag new elements. This can be addressed as an optimization problem, namely, given a category, finding the best partition of attributes that maximizes the coverage and precision for each attribute. We leave this task for future work.

E. Heterogeneous categories

Heterogeneous categories fall outside the of the work presented here, as stated in Definition 3.1. However, we would like to quickly discuss what happens if we apply this methodology to heterogeneous categories.

One of the categories not listed above is Baby Carriers. For this category we obtain a precision of 85.15%. However, if we go a category up in the taxonomy, Baby Goods, the precision goes down to 63.16%. The category Baby Goods is quite heterogeneous, containing also baby clothes, toys, etc. There is a plethora of semantically different attributes for each of these categories with often overlapping values, rendering the model imprecise.

IX. Conclusions and Future Work

In this paper we describe a successful end-to-end semi-supervised approach that automatizes the task of obtaining attributes and values for products in an e-commerce environment. It applies a semi-supervised approach to bootstrap an initial seed of labeled instances obtained automatically from HTML tables in product pages.

The approach is domain- and language-independent (except the tokenizer and part-of-speech tagger) and requires little to no human intervention. Although it has been applied to e-commerce, it can be applied to any domain where tables can provide some semantic information $<$*attributes, value*$>$ to automatically extract the starting seed.

We have presented an extensive quantitative and qualitative evaluation across simple and complex values, different categories and languages, with thousands of triples manually evaluated. We show that the approach obtains good coverage while maintaining high precision, which is the main business need. This is attained by combining a bootstrap-based core with different methods that specifically target various known problems of bootstrapping, namely semantic drift, wrong name entity boundaries, and lack of generalization in the training set.

We discuss the variations in performance for different languages (Japanese and German) and ML algorithms (CRF and RNN). The results obtained for the two languages are comparable. Regarding the ML algorithm, they both produce good results. However, the worst case in CRF is better than the worse case in LSTM. They often make similar mistakes, but they can complement each other.

We show that the number and type of attributes selected to create the model has a major impact on the performance of the system. For some attributes, using one global model is good enough, but for the least performing attributes we need to fine tune the attribute set given to the tagger. We also show that, although the category itself is not so relevant, the homogeneity of the category is.

Future work includes improving the machine learning model by combining different approaches, partitioning the attributes to obtain better precision/coverage, homogenizing attribute values, and extending this evaluation to other languages and domains outside e-commerce.

References

[1] Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, volume 7, pages 2670–2676, 2007.

[2] Lidong Bing, Tak-Lam Wong, and Wai Lam. 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.

[3] Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110. ACM, 2010.

[4] Bruno Charron, Yu Hirate, David Purcell, and Martin Rezk. *Extracting Semantic Information for e-Commerce*, pages 273–290. Springer International Publishing, Cham, 2016.

[5] Hsin-Hsi Chen, Shih-Chung Tsai, and Jin-He Tsai. Mining tables from large scale html texts. In *Proceedings of the 18th Conference on Computational Linguistics*, pages 166–172. Association for Computational Linguistics, 2000.

[6] Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110, 1999.

[7] James R. Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with mutual exclusion bootstrapping. In *In Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180, 2007.

[8] Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102. Association for Computational Linguistics, 2017.

[9] Franck Dernoncourt, Ji Young Lee, Ozlem Uzuner, and Peter Szolovits. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical Informatics Association (JAMIA)*, 2016.

[10] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134, June 2005.

[11] Rahul Gupta, Alon Halevy, Xuezhi Wang, Steven Whang, and Fei Wu. Biperpedia: An ontology for search applications. In *Proc. 40th Int'l Conf. on Very Large Data Bases (PVLDB)*, 2014.

[12] Sonal Gupta and Christopher D Manning. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108, 2014.

[13] Ralf Krestel, René Witte, and Sabine Bergler. Predicate-Argument EXtractor (PAX). In *New Challenges for NLP Frameworks*, pages 51–54. ELRA, 2010.

[14] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[15] Colin Lockard, Xin Luna Dong, Arash Einolghozati, and Prashant Shiralkar. Ceres: Distantly supervised relation extraction from the semi-structured web. *Proc. VLDB Endow.*, 11(10):1084–1096, June 2018.

[16] Tom M. Mitchell, William W. Cohen, Estevam R. Hruschka Jr., Partha Pratim Talukdar, Justin Betteridge, Andrew Carlson, Bhavana Dalvi Mishra, Matthew Gardner, Bryan Kisiel, Jayant Krishnamurthy, Ni Lao, Kathryn Mazaitis, Thahir Mohamed, Ndapandula Nakashole, Emmanouil Antonios Platanios, Alan Ritter, Mehdi Samadi, Burr Settles, Richard C. Wang, Derry Tanti Wijaya, Abhinav Gupta, Xinlei Chen, Abulhair Saparov, Malcolm Greaves, and Joel Welling. Never-ending learning. In *AAAI, Texas, USA.*, pages 2302–2310, 2015.

[17] Marius Pasca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: Fact extraction in the fast lane. In *ACL*, 2006.

[18] Petar Petrovski and Christian Bizer. Extracting attribute-value pairs from product specifications on the web. In *Proceedings of the International Conference on Web Intelligence*, WI '17, pages 558–565, New York, NY, USA, 2017. ACM.

[19] Duangmanee Pew Putthividhya and Junling Hu. Bootstrapped named entity recognition for product attribute extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567. Association for Computational Linguistics, 2011.

[20] Disheng Qiu, Luciano Barbosa, Xin Luna Dong, Yanyan Shen, and Divesh Srivastava. DEXTER: Large-Scale Discovery and Extraction of Product Specifications on the Web. *PVLDB*, 8(13):2194–2205, 2015.

[21] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3567–3575. Curran Associates, Inc., 2016.

[22] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[23] Ellen Riloff and Rosie Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, AAAI '99/IAAI '99, pages 474–479, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.

[24] Keiji Shinzato and Satoshi Sekine. Unsupervised extraction of attributes and their values from product description. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013*, pages 1339–1347, 2013.

[25] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

[26] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. Opentag: Open attribute value extraction from product profiles. In *KDD*, pages 1049–1058. ACM, 2018.