Extracting Multilingual Natural-Language Patterns for RDF Predicates

Daniel Gerber and Axel-Cyrille Ngonga Ngomo

Universität Leipzig, Institut für Informatik, AKSW,
Postfach 100920, D-04009 Leipzig, Germany,
{dgerber|ngonga}@informatik.uni-leipzig.de
http://aksw.org

Abstract. Most knowledge sources on the Data Web were extracted from structured or semi-structured data. Thus, they encompass solely a small fraction of the information available on the document-oriented Web. In this paper, we present BOA, a bootstrapping strategy for extracting RDF from text. The idea behind BOA is to extract natural-language patterns that represent predicates found on the Data Web from unstructured data by using background knowledge from the Data Web. These patterns are used to extract instance knowledge from natural-language text. This knowledge is finally fed back into the Data Web, therewith closing the loop. The approach followed by BOA is quasi independent of the language in which the corpus is written. We demonstrate our approach by applying it to four different corpora and two different languages. We evaluate BOA on these data sets using DBpedia as background knowledge. Our results show that we can extract several thousand new facts in one iteration with very high accuracy. Moreover, we provide the first multilingual repository of natural-language representations of predicates found on the Data Web.

1 Introduction

The population of the Data Web has been mainly carried out by transforming semi-structured and structured data available on the Web into RDF. Yet, while these approaches have successfully generated the more than 30 billion triples currently available on the Linked Open Data Cloud [2], they rely on background data that encompasses solely 15-20% [3, 9] of the information on the Web, as the rest of the information in the document-oriented Web is only available in unstructured form. Consequently, the data in the Linked Data Web suffers from a lack of coverage and actuality that has been eradicated from the Web by Web 2.0 and crowdsourcing approaches.

In this paper, we extend on the BOA framework [1] as presented in [10]. The goal of the BOA framework is to allow extracting structured data as RDF from unstructured data. Unlike many approaches (e.g., [4]) which start with

[1] A demo of the framework can be found at http://boa.aksw.org. The code of the project is at http://boa.googlecode.com

their own ontologies and background knowledge as seeds, BOA makes use of the vast amount of knowledge available on the Linked Data Web to retrieve high-confidence natural-language patterns that express the predicates available in the Data Web. The recognition of high-confidence patterns is carried out by using a supervised machine learning trained on a small set of manually annotated patterns. Based on these patterns, BOA can extract new instance knowledge (i.e., both new entities and relations between these new entities) from the Human Web with high accuracy. Our approach is completely agnostic of the knowledge base upon which it is deployed. It can thus be used on the whole Data Web. In addition, BOA implements generic pattern extraction algorithms that can be used to retrieve knowledge from sources written in different languages. Consequently, it can also be used on the whole Human Web. One of the byproducts of the BOA framework is a set of natural-language patterns that constitutes a valuable resource for tools that require the conversion of natural language into structured data, (e.g., semantic information retrieval [22] and question answering[2] frameworks).

The main contributions of this paper are as follows:

- We present the approach implemented by the BOA framework and apply it to corpora written in English and in German.
- We provide a multilingual library of natural-language representations of predicates found on the Data Web (especially in DBpedia).
- We present a set of features that can be used to distinguish high-quality from poor natural-language patterns for Data Web predicates.
- We evaluate our machine-learning approach and the BOA framework on 4 text datasets against DBpedia and show that we can achieve a high-accuracy extraction in both languages.

The rest of this paper is structured as follows: In Section 2, we give an overview of previous work that is related to our approach. Thereafter, in Section 3, we present our bootstrapping framework and several insights that led to the approach currently implemented therein. In Section 4 we evaluate our approach on two different data sets and show its robustness and accuracy. Finally, we discuss our results and conclude.

2 Related Work

BOA is related to a large number of disciplines due to the different areas of knowledge from which it borrows methods. Like Information Extraction approaches, BOA aims to detect entities in text. Three main categories of natural language processing (NLP) tools play a central role during the extraction of knowledge from text: Keyphrase Extraction (KE) algorithms aim to detect multi-word units that capture the essence of a document [14, 13]. Named Entity Recognition (NER) approaches try to discover instances of predefined classes of

[2] `http://autosparql-tbsl.dl-learner.org`

entities [20, 8]. Finally, Relation Extraction (RE) approaches are used to discover the relations between the entities detected by using NER [16, 23]. While these three categories of approaches are suitable for the extraction of facts from NL, the use of the Data Web as source for background knowledge for fact extraction is still in its infancy. [16] coined the term "distant supervision" to describe this paradigm but developed an approach that led to extractors with a low precision (approx. 67.6%). The most precise approaches for RE rely on supervised machine learning [19, 24, 5, 7]. Thus, they can only extract a small fraction of the knowledge on the Web due to the scarcity of large training datasets.

In addition to the work done by the NLP community, several frameworks have been developed with the explicit purpose of bridging the gap between NLP and the Data Web by extracting RDF and RDFa out of NL [11, 1]. Services such as Alchemy[3], OpenCalais[4], FOX [18] and Spotlight [15] allow to extract entities and relations from text. Yet, they do not rely on the Data Web as training data and are thus restricted with respect to the number of relations they can detect.

The problem of extracting knowledge from the Web at large scale, which is most closely related to this paper, has been the object of recent research, especially in the projects ReadTheWeb and PROSPERA. The aim of the ReadTheWeb project[5] [4] is to create the never-ending language learner NELL that can read webpages. To achieve this goal, NELL is fed with the ClueWeb09[6] data set (1 billion web pages, 10 languages, approximately 5TB) cyclically. The input data for NELL consisted of an initial ontology that contained hundreds of categories and relations as well as a small number of instances for each category and relation. In each iteration, NELL uses the available instance knowledge to retrieve new instances of existing categories and relations between known instances by using pattern harvesting. The approach followed by PROSPERA [17] is similar to that of NELL but relies on the iterative harvesting of n-grams-itemset patterns. These patterns allow to generalize NL patterns found in text without introducing more noise into the patterns during the generalization process. In addition, PROSPERA uses reasoning to discard statements that are logically inconsistent with the available knowledge.

Our approach goes beyond the state of the art in two key aspects. First, it is the first approach to extract multi-lingual natural-language patterns from the Linked Data Web. In addition, it makes use of the Data Web as background knowledge, while the approaches ReadTheWeb and PROSPERA rely on a their own ontology for this purpose. Moreover, the results of most other approaches cannot be integrated directly into the Data Web. In contrast, BOA generates RDF and can thus be used to populate a knowledge base that can be readily made available for querying via SPARQL, integrating and linking. Finally, our experiments show that our approach can extract a large number of statements (like PROSPERA and [16]) with a high precision (like ReadTheWeb). For

[3] `http://www.alchemyapi.com`
[4] `http://www.opencalais.org`
[5] `http://rtw.ml.cmu.edu`
[6] `http://lemurproject.org/clueweb09`

example, 78,944 of the 80,773 statements extracted by BOA from the English Wikipedia[7] where not available in DBpedia.

3 Approach

In this section, we present the BOA framework. We begin by giving an overview of the architecture it implements. Thereafter, we give a deeper presentation of its core components. We begin by explicating the pattern extraction process. Then, we focus especially on the features we extract while searching for adequate patterns. We present our use of neural networks for learning a score function. Finally we show how the scored patterns are used to generate RDF.

3.1 Overview

The idea behind the BOA architecture was to provide an architecture that allows extracting structured data from the Human Web iteratively. An overview of the workflow implemented by BOA is given in Figure 1. The input for the BOA framework consists of a set of knowledge bases, a text corpus (mostly extracted from the Web) and (optionally) a Wikipedia dump[8]. When provided by a Wikipedia dump, the framework begins by generating surface forms for all entities in the source knowledge base. The surface forms used by BOA are generated by using an extension of the method proposed in [15]. For each predicate p found in the input knowledge sources, BOA carries out a sentence-level statistical analysis of the co-occurrence of pairs of labels of resources that are linked via p. Instead of using a hard-coded evaluation function like in previous work, BOA then uses a supervised machine-learning approach to compute the score of patterns for each combination of corpus and knowledge bases. In a final step, our framework uses the best-scoring patterns for each relation to generate RDF data. This data and the already available background knowledge can now be used for a further iteration of the approach. In the following, we describe the core steps of BOA in more detail. Throughout this description, we will use the example of generating new knowledge for the `dbpedia:architect` relation.

3.2 Pattern Extraction

Let $\mathcal{K}$ be the knowledge base that is used as background knowledge. The first and optional step of the pattern extraction is the computation of surface forms $\mathcal{S}_r$ for the subject and objects of a relation p for which patterns are to be extracted. To extract surface forms for resources $r \in \mathcal{K}$, we use Wikipedia's redirect and disambiguation pages as described in [15]. (see Table 1 for a statistical overview of the surface forms for resources found in DBpedia). The main drawback of this approach is that it is not tailored towards the extraction of datatype properties.

[7] `http://wikipedia.org`
[8] `http://wikipedia.c3sl.ufpr.br/`

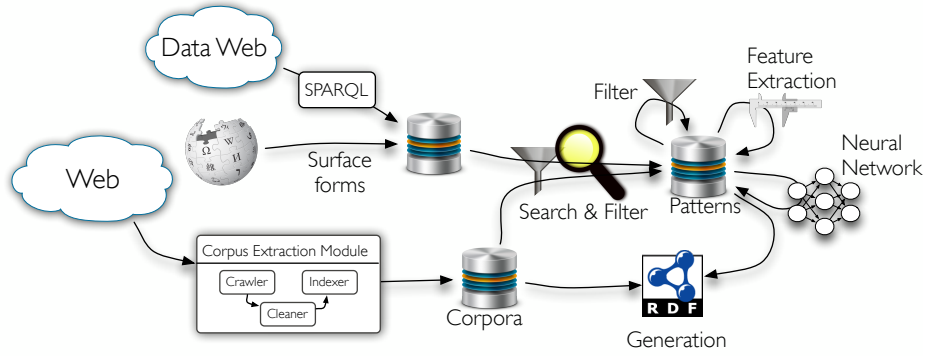

Fig. 1. Overview of the BOA approach.

Consequently, if the object of a relation is a datatype (i.e., not a resource), we use data generation modules that allow parsing the datatype at hand and generating possible surface forms for it. For example, when provided with the date "11/12/2012", BOA generates strings such as "11th Dec 2012", "11th December 2012" and "Dec 11, 2012". By using surface forms, we achieve a significantly higher recall than without as shown in Figure 2. It is important to note that the use of Wikipedia dumps does not limit our framework as they exist in more than 90 languages. In addition, BOA can also run without being given surface forms.

	German	English
Number of URIs	1.280.859	3.496.082
Number of all surface forms	2.124.084	8.252.275
Maximum of surface forms per resource	132	981
Average of surface forms per resource	1,658	2,360

Table 1. Statistical overview of German and English surface forms.

The pattern search is carried out independently for each predicate. Let $\mathtt{p} \in \mathfrak{P}$ be a predicate whose natural-language representations are to be detected, where $\mathfrak{P}$ is the set of all predicates. We use the symbol "$\in$" between triples and knowledge bases to signify that a triple can be found in a knowledge base. The starting point for the pattern search for $\mathtt{p}$ is the set of pairs $\mathcal{I}(\mathtt{p}) = \{(s, o) : (s\ \mathrm{p}\ o) \in \mathcal{K}\}$ that instantiate $\mathtt{p}$. In the following, we use $\lambda(x)$ to signify the set of labels of any resource x and $\mu(x)$ to signify x's URI. The pattern search process begins with the even distribution of the set $\mathcal{I}(\mathtt{p})$ across pattern search threads. Each of these threads then retrieves all sentences which contain both labels of all combination of $(\lambda(s), \lambda(o))$ from the input corpus. If a thread finds

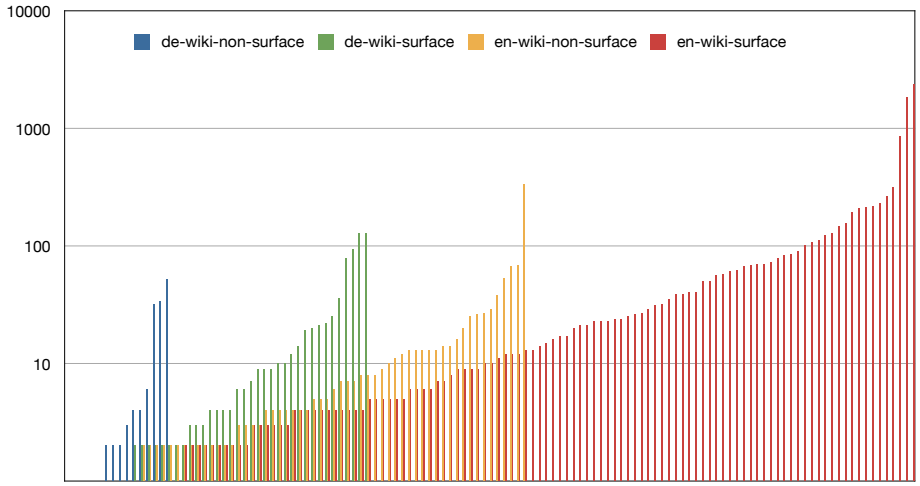

Fig. 2. Distribution of patterns per pattern mapping in logarithmic scale. Each vertical line represents one pattern mapping.

a sentence σ that contains a label $l_s \in \lambda(s)$ and $l_o \in \lambda(o)$, it deletes all tokens that are not found between l_s and l_o in σ. The labels are then replaced with the placeholders ?D? for l_s and ?R? for l_o. We call the resulting string a *natural-language representation* (NLR) of $\mathtt{p}$ and denote it with θ. Each θ extracted is used to create a new instance of a BOA pattern.

Definition 1 (BOA Pattern). *A BOA pattern is a pair* $\mathcal{P} = (\mu(\boldsymbol{p}), \theta)$*, where* $\mu(\boldsymbol{p})$ *is* $\boldsymbol{p}$*'s URI and* θ *is a natural-language representation of* $\boldsymbol{p}$*.*

Definition 2 (BOA Pattern Mapping). *A BOA pattern mapping is a function* $\mathcal{M}$ *such that* $\mathcal{M}(\boldsymbol{p}) = \mathfrak{S}$ *, where* $\mathfrak{S}$ *is the set of natural-language representations for* $\boldsymbol{p}$*.*

```
1 dbr:Empire_State_Building dbo:architect dbr:Shreve,_Lamb_and_Harmon
2 dbr:Empire_State_Building rdfs:label 'Empire State Building'@en
3 dbr:Shreve,_Lamb_and_Harmon rdfs:label 'Shreve, Lamb and Harmon'@en
```

Listing 1. RDF snippet used for pattern search

For example, consider the RDF snippet from Listing 1 derived from DBpedia. Querying the index of an underlying corpus for sentences which contain both entity labels returns the sentences depicted in Table 2 amongst others. We can replace "Empire State Building" with ?D?, because it is a label of the subject of the `:architect` triple, as well as replace "Shreve, Lamb and Harmon" and "William F. Lamb" (a surface form of l_r) with ?R? because it is one label of the object of the same triple. These substitutions lead to the BOA patterns (`:architect`, "?D? *was designed by* ?R?") and (`:architect`, "?R? *also designed*

the ?D?"). For the sake of brevity and in the case of unambiguity, we also call θ "pattern". Patterns are only considered for storage and further computation

Sentence with $\lambda(s)$ before $\lambda(o)$	Sentence with $\lambda(o)$ before $\lambda(s)$
"... **Shreve, Lamb and Harmon** *also designed the* **Empire State Building** ."	"The **Empire State Building** *was designed by* **William F. Lamb** ..."

Table 2. Example sentences for pattern search.

if they withstand a first filtering process. For example, they must contain more than one non stop-word, have a token count between certain thresholds and may not begin with "and" or ", and". In addition to $\mathcal{M}(\mathtt{p})$ for each $\mathtt{p}$, we compute compute the number $f(\mathcal{P}, s, o)$ of occurrences of $\mathcal{P}$ for each element (s, o) of $\mathcal{I}(\mathtt{p})$ and the ID of the sentences in which $\mathcal{P}$ was found. Based on this data, we can also compute

- the total number of occurrences of a BOA pattern $\mathcal{P}$, dubbed $f(\mathcal{P})$;
- the number of sentences that led to θ and that contained $\lambda(s)$ and $\lambda(o)$ with $(s, o) \in \mathcal{I}(\mathtt{p})$, which we denote $l(s, o, \theta, \mathtt{p})$ and
- $\mathcal{I}(\mathtt{p}, \theta)$ is the subset of $\mathcal{I}(\mathtt{p})$ which contains only pairs (s, o) that led to θ.

Thereafter we apply a second filtering process, where patterns which do not abide a threshold for $|\mathcal{I}(\mathtt{p}, \theta)|$, $\max(l(s, o, \theta, \mathtt{p}))$ and $f(\mathcal{P})$ are removed. We denote the set of predicates such that the pattern $\theta \in \mathcal{M}(\mathtt{p})$ by $\mathfrak{M}(\theta)$. Note that pattern mappings for different predicates can contain the same pattern.

3.3 Feature Extraction

The feature extraction is applied on all patterns which overcome both filtering processes. The pattern mappings are evenly distributed to feature extraction threads. Each thread then computes the features for all patterns of the given pattern mappings. Note that although BOA is designed to work independently from the language of the underlying corpus, it can be tailored towards a given language. For example the ReVerb and IICM feature exploit knowledge that is specific to English.

Support The support features $s_1(\theta, \mathtt{p})$ and $s_2(\theta, \mathtt{p})$ of the pattern θ for the predicate $\mathtt{p}$ captures how often a pattern occurs between the elements of $\mathcal{I}(\mathtt{p})$. Especially, $s_1(\theta, \mathtt{p})$ stands for the number of distinct pairs the pattern has been learned from, while $s_2(\theta, \mathtt{p})$ is the maximum number of occurrences of the same pattern between the elements of a single element of $\mathcal{I}(\mathtt{p})$. Formally,

$$s_1(\theta, \mathtt{p}) = \log(|\mathcal{I}(\mathtt{p}, \theta)|) \qquad s_2(\theta, \mathtt{p}) = \log\left(\max_{(s,o)\in\mathcal{I}(\mathtt{p})} l(s, o, \theta, \mathtt{p})\right)$$

Since both components of the support are Poisson-distributed (see [10]), we use the logarithm to reduce the boosting of very popular patterns.

Specificity A pattern θ is considered to be specific when it occurs in a small number of pattern mappings, i.e., when it expresses exclusively `p`. We adapted the idea of inverse document frequency (*idf*) as known from Information Retrieval to capture this characteristic. The specificity $i(\theta)$ of θ is thus given by the following expression:

$$i(\theta) = \log\left(\frac{|\mathfrak{P}|}{|\mathfrak{M}(\theta)|}\right).$$

Typicity A pattern θ is considered to display a high typicity with respect to a predicate `p` if it connects only entity labels with match the range and domain restrictions of `p`. Let d resp. r be functions that map each `p` to the highest super-class (except *owl:Thing*) of its `rdfs:domain` resp. `rdfs:range` in the provided ontology. Furthermore, let $domain(\theta, \sigma)$ resp. $range(\theta, \sigma)$ be functions which map the class of the named entity used to substitute ?D? resp. ?R? in the pattern θ for the given sentence σ. Finally, let the function $\delta(x, y)$ be a function that returns 1 if $x = y$ and else 0. We define the typicity features of θ as

$$t_1(\theta, p) = \sum_{\sigma \in S} \frac{\delta(d(p), domain(\theta, \sigma))}{|S|} \qquad t_2(\theta, p) = \sum_{\sigma \in S} \frac{\delta(r(p), range(\theta, \sigma))}{|S|}$$

and $t_3(\theta, p) = \log(|S| + 1)$, where S is the set of sentences used to evaluate the typicity of θ. Note that t_1 and t_2 of the typicity features are simply the precision of the pattern with respect to domain and range. We use t_3 to prevent overly promoting patterns which have a low recall, i.e., patterns that return only a small number of sentences. Also note, that the detection of $domain(\theta, \sigma)$ resp. $range(\theta, \sigma)$ is a demanding task, which we solved so far by using a trained NER tagger.

IICM The **I**ntrinsic **I**nformation **C**ontent **M**etric (IICM) captures the semantic relatedness between a pattern's NLR and the property it expresses. This similarity measure has been introduced in [21] and is based in the Jiang-Conrath similarity measure [12]. The advantage of this measure is that it is purely based on the hierarchical structure of words inside an ontology like WordNet, thus avoiding heavyweight corpus statistics. We apply this measure to each BOA pattern mapping independently. First we retrieve all synsets for each token of the pattern mappings associated *rdfs:label* from WordNet. For the pattern mapping $\mathcal{M}$(`dbpedia-owl:architect`) this is "architect" and "designer". If no such synsets are found we use the tokens of the *rdfs:label* of $\mathcal{M}$(`p`). We then apply the IICM measure pairwise to these tokens and the tokens derived from one $\mathcal{M}$(`p`) assigned pattern's NLR. The IICM score for one pattern is then the maximum value of all pairs similarity scores.

ReVerb ReVerb has been introduced in [6] and distinguishes good from bad relation phrases by two simple constraints. The first constraint is of syntactical nature and is a part-of-speed-based regular expression, where a token sequence is considered to be good instance of a relation if it contains a verb. The second constraint is based on the intuition that a valid relation phrase should take many distinct arguments in a large corpus, therewith avoiding the selection of overly specific relation phrases. Since the input of ReVerb is

a POS-tagged sentence, but a pattern is only a substring of a sentence, we use all sentences we found the pattern in (see Section 3.2) as ReVerbs input. For all of ReVerb's extracted relations of a particular sentence we check if it matches the pattern in question and use ReVerb's trained logistic regression classifier to assign a confidence score to this extraction. Note that BOA focuses on the relation between two given resources and discards all other extractions, since those are not mappable to the background knowledge. Finally, we calculate a patterns ReVerb feature as the average of all scored extractions.

Tf-Idf The Tf-Idf features are an adaption of the tf-idf score used in information retrieval and text mining. The intuition behind this feature, as in information retrieval, is to distinguish relevant from irrelevant patterns for a given pattern mapping $\mathcal{M}(\mathtt{p})$. In the BOA case a document is considered to be all tokens of all patterns (without stop-words and the placeholders "?D?" and "?R?") of one pattern mapping. In other words, to total number of documents is equal to the number of pattern mappings with patterns. We then calculate the features $idf(\mathtt{p})$ and $tf(\mathtt{p})$ for each token of the patterns NLR as follows:

$$idf(\mathtt{p}) = \sum_{t \in \mathcal{T}(\mathtt{p})} log(\frac{|\mathcal{M}(\mathtt{p})|}{df(t)+1}) + 1 \qquad tf(\mathtt{p}) = \sum_{t \in \mathcal{T}(\mathtt{p})} \sqrt{f(t)}$$

Where $df(t)$ is the document frequency of t, $f(t)$ the term frequency of t and $\mathcal{T}(\mathtt{p})$ the set of tokens for a pattern $\mathtt{p}$.

3.4 Scoring Approach

Given the number of features that characterize the input data, devising a simple scoring function transforms into a very demanding task. In this work, we address the problem of computing a score for each BOA pattern by using feedforward neural networks. The input layer of our network consists of as many neurons as features for patterns while the output neuron consists of exactly one neuron whose activation was used as score. We used the sigmoid function as transfer function. For each data set, we trained the neural network by using manually annotated patterns (200 in our experiments). The patterns were extracted from the set of all patterns generated by BOA by first randomly sampling the same number of patterns for each predicate (7 in our experiments) and selecting a subset of these patterns for annotation.

3.5 RDF Generation

The generation of RDF out of the knowledge acquired by BOA is the final step of the extraction process. When using BOA iteratively, the output of each RDF generation would provide parts of the input for the subsequent extraction process. In previous work, semantic drift has been shown to be one of the key problems of such iterative approaches [4, 17]. In order to maintain a high precision and to avoid semantic drift within the BOA framework, we solely select

the top-n percent of all scored patterns for generating RDF. As our evaluation shows, this approach is sufficient to avoid selecting noisy patterns. All patterns which abide by these two conditions are used to retrieve sentences that can be used for RDF generation.

The RDF generation *per se* is carried out as follows: For each pattern θ and each predicate `p`, we first use the Lucene index to retrieve sentences that contain θ stripped from the placeholders "?D?" and "?R?". These sentences are subsequently processed by a NER tool that is able to detect entities that are of the `rdfs:domain` and `rdfs:range` of `p`. Thereafter, the first named entities within a limited distance on the left and right of θ which abide by the domain and range restrictions of `p` are selected as labels for subject and object of `p`. Each of the extracted labels is then fed into the URI retrieval and disambiguation service implemented by the FOX framework[9]. If this service returns a URI, then we use it for the label detected by BOA. Else, we create a new BOA URI.

Once we have computed URIs, we are able to generate RDF triples. The labels found in the text corpus are attached to the URI by using `rdfs:label`. In addition, note that we are even able to add `rdf:type` statements to our knowledge base by utilizing the domain and range restrictions of `p`. Note as well that it is possible, even desirable that one triple is found by multiple patterns. We use information to calculate a confidence scores $s(t)$ for each triple t that we extract:

$$s(t) = \frac{1}{1 + e^{-\left[\sum_{i=1}^{n} s(p_i(t))\right] n+1}}$$

where $\sum s(p_i(t))$ is the sum of the score given by the neural network of all patterns that found the triple t and n the number of all This function is derived from the sigmoid function which outputs values ranging from 0 to 1 inclusively. We modified the function to boost patterns which are learned by more than one pattern. The triple score is the sum of the patterns scores which found the triple multiplied by the number of triples. With the help of this score, we can avoid semantic drift in further iterations by solely selecting the top n percent of the triples to use as background knowledge in the next iteration. By applying our approach, we were able to extract triples such as `dbr:Washington_Monument dbo:architect dbr:Robert_Mills`, which is not in DBpedia but explicitly stated in Wikipedia. The results of our extraction can be explored via the dashboard shown in Figure 3.

4 Evaluation

4.1 Experimental Setup

The aim of BOA's evaluation was three-fold. First, we aimed at testing whether BOA can be really applied to different languages. To achieve this goal, we applied BOA to German and English. Our second goal was to determine the accuracy

[9] A demo of the framework is available at `http://fox.aksw.org`

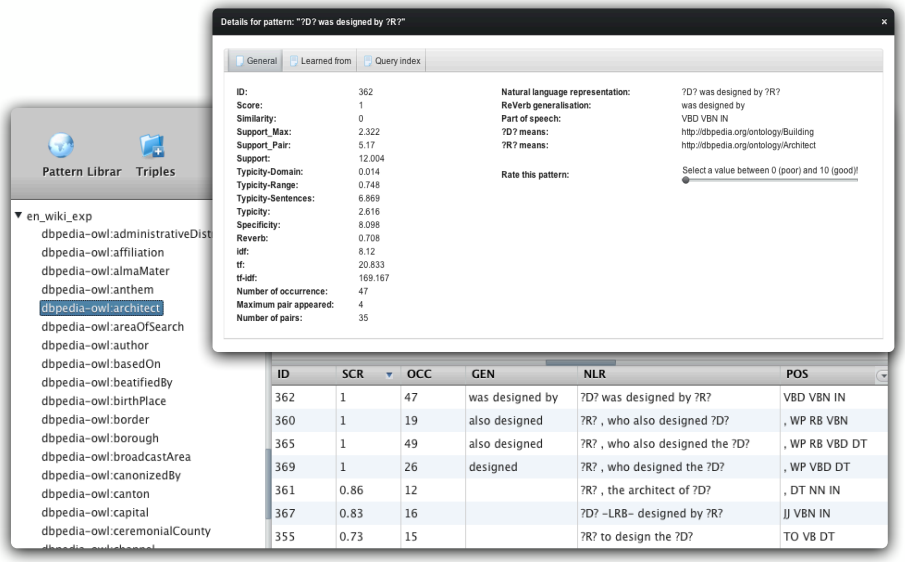

Fig. 3. The BOA pattern library GUI.

of BOA's extraction. For this purpose, we sample 100 triples from the data extracted by BOA from each corpus and had two annotators measure the precision of these samples manually. Finally, we wanted to compute the amount of (new) knowledge that can be extracted by BOA. For this purpose, we compute the number of new triples that we were able to extract. We excluded temporal properties from the evaluation as BOA does net yet distinguish between different time expressions and conjugations. We evaluated our approach on 4 corpora written in German and English. The first two corpora, en-wiki resp. de-wiki, were extracted from the English resp. German Wikipedia dumps and contained 58 million resp. 24.6 million sentences. The other two corpora (en-news resp. de-news) were extracted from English resp. German news corpora and were significantly larger, containing, 241.3 resp. 112.8 sentence.

4.2 Score Function

We began the evaluation by annotating 200 patterns per corpus by hand. Each training data set was annotated independently by the authors, who agreed on the annotations in approximately 90% of the cases. The annotations upon which the authors disagreed were resolved by both authors. High-quality patterns were assigned a score of 1, else they were assigned a 0. We then trained four different neural networks (one for each dataset) to distinguish between the high-precision and poor patterns. In out experiments, We varied the size of the hidden layer between one and three times the size of the input layer. In addition, we varied the

error rate to which they were trained. The maximal number of training epochs was set to 10000. The accuracy of the networks was measured by using a 10-fold cross-validation. Patterns scored to be above 0.5 were considered to be good patterns, while all other were considered to be poor. The best neural network was set to be the smallest network that reaches the maximal accuracy. The resulting learning curves are shown in Figure 4. It is easy to see that networks trained to achieve an error rate of maximally 5% performed best in our experiments.

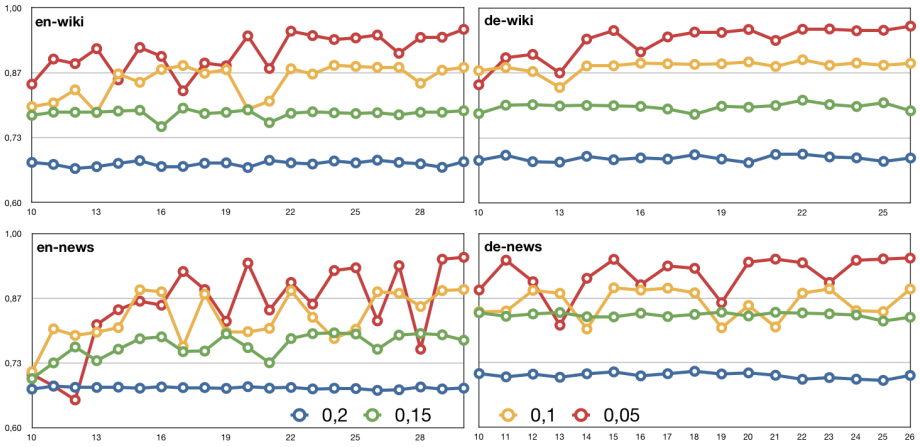

Fig. 4. Learning curves of BOA's neural networks.

4.3 Multi-Linguality

Enabling BOA to process languages other than English require solely the alteration of the NER tools and POS parsers. As the results on German show, languages with a wide range of morpho-syntactical variations demand the analysis of considerably larger corpora to enable the detection of meaningful patterns. For example, while we trained the neural network by using the same number of patterns, we were not able to detect any triples with a score above 0.5 when using the German Wikipedia and DBpedia. Yet, when using a larger German Newscorpus data set, we were able to detect new patterns with an accpetable precision (see subsequent section).

4.4 Accuracy

We measured the precision of the extraction carried out by BOA as well as the number of new triples that we were able to extract in one iteration. We achieve a precision superior to 90% overall on the English data sets. This value

is comparable to that achieved by the previous versions of BOA [10]. Yet, the addition of surface forms for the extraction yields the advantage of achieving a considerably higher recall both with respect to the number of patterns extracted as well as with respect to the total number of triples extracted. For example, when using the English Wikipedia, we can extract approximately more than twice the amount of triples. The same holds for the number of patterns and pattern mappings as shown in Figure 2.

	en-wiki	**de-wiki**	**en-news**	**de-news**
Number of pattern mappings	125	44	66	19
Number of patterns	9551	586	7366	109
Number of new triples	78944	22883	10138	883
Number of known triples	1829	798	655	42
Number of found triples	80773	3081	10793	925
Precision Top-100	92%	70%	91%	74%

Table 3. Results of the first iteration of the BOA framework.

An excerpt of the new knowledge extracted by BOA is shown in Listing 2. Note that the triple `Iomega subsidiary ExcelStor_Technology` is wrong. Although Iomega planned to buy ExcelStor, the deal was never concluded. Our approach finds the right patterns in the sentences describing the temptative deal and thus extract this triple.

```
Chinese spokenIn Malaysia .
Chinese spokenIn  China .
Italian spokenIn  Italy .
Greek spokenIn United_States .

ESV_Blau-Rot_Bonn ground Bonn
TG_Würzburg ground Würzburg

Weitnau administrativeDistrict boa:Oberallgäu .
Memmingerberg administrativeDistrict boa:Unterallgäu .
Borsdorf administrativeDistrict Leipzig .
Wirsberg administrativeDistrict Kulmbach .

IBM subsidiary boa:Softek_Storage_Solutions .
Cerberus_Capital_Management subsidiary boa:Chrysler_Holdings_LLC .
Kaspersky_Lab subsidiary boa:Spamtest_Project .
Intel_Corporation subsidiary McAfee .
Iomega subsidiary ExcelStor_Technology

Pierre-Joseph-Olivier_Chauveau party Conservative_Party_of_Quebec .
Robert_Stanfield party Progressive_Conservative_Party .
Adélard_Godbout party Quebec_Liberal_Party .

American_Airlines hubAirport Logan_International_Airport .
Frontier_Airlines hubAirport Akron-Canton_Regional_Airport .
El_Al hubAirport  Los_Angeles_International_Airport

Neil_Warnock managerClub Sheffield_United_F.C. .
boa:Akira_Ogi managerClub Orix_Buffaloes .
Richard_Money managerClub Walsall .

Walter_Albrecht_Becker birthPlace Hamburg .
```

```
Sophie_Ludovike_Reichenbach birthPlace Ludwigsburg .
Hans_Dreier birthPlace Bremen .
```

Listing 2. RDF extracted by BOA. If not stated otherwise, all instances and properties use the DBpedia namespace.

5 Conclusion and Future Work

In this paper, we presented BOA, a framework for the extraction of RDF from unstructured data. We presented the components of the BOA framework and applied it to English and German corpora. We showed that in all cases, we can extract RDF from the data at hand. Our extraction strategy was to only integrate RDF triples that were generated by at least two patterns. By these means we were able to achieve a high precision on all data sets. The precision of German was smaller than that on English because of the rich morphology and syntax of the German language. Overall, the new version of BOA achieves a significantly higher recall by two means. First, we used surface forms to retrieve entities. In addition, we devised an approach to extract triples from datatype properties. In future work, we will aim to apply our approach to Asian languages whose grammar differ completely from that of the language we processed so far. In addition, we will consider the use of crowd-sourcing to improve our scoring approach. Finally, we will take temporal markers into consideration so as to be able to process predicates such as `dbpedia:formerTeam`.

References

1. B. Adrian, J. Hees, I. Herman, M. Sintek, and A. Dengel. Epiphany: Adaptable RDFa Generation Linking the Web of Documents to the Web of Data. In *EKAW*, pages 178–192, 2010.
2. S. Auer, J. Lehmann, and A.-C. N. Ngomo. Introduction to linked data and its lifecycle on the web. In *Reasoning Web*, pages 1–75, 2011.
3. R. Blumberg and S. Atre. The problem with unstructured data. *DM Review*, 13(February 2003):42–49, 2003.
4. A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
5. J. R. Curran and S. Clark. Language independent ner using a maximum entropy tagger. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003 - Volume 4*, pages 164–167, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
6. A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545. ACL, 2011.
7. J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, ACL '05, pages 363–370, 2005.
8. J. R. Finkel and C. D. Manning. Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data. In *ACL '10*, pages 720–728, 2010.

9. A. Gaag, A. Kohn, and U. Lindemann. Function-based solution retrieval and semantic search in mechanical engineering. In *IDEC '09*, pages 147–158, 2009.
10. D. Gerber and A.-C. Ngonga Ngomo. Bootstrapping the linked data web. In *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011.
11. D. Huynh, S. Mazzocchi, and D. R. Karger. Piggy bank: Experience the semantic web inside your web browser. In *ISWC*, pages 413–430, 2005.
12. J. J. Jiang and D. W. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, pages 9008+, Sept. 1997.
13. S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *SemEval '10*, pages 21–26, 2010.
14. Y. Matsuo and M. Ishizuka. Keyword Extraction From A Single Document Using Word Co-Occurrence Statistical Information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.
15. P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In C. Ghidini, A.-C. N. Ngomo, S. N. Lindstaedt, and T. Pellegrini, editors, *I-SEMANTICS*, ACM International Conference Proceeding Series, pages 1–8. ACM, 2011.
16. M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. *ACL*, pages 1003–1011, 2009.
17. N. Nakashole, M. Theobald, and G. Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, Hong Kong, 2011.
18. A.-C. Ngonga Ngomo, N. Heino, K. Lyko, R. Speck, and M. Kaltenböck. SCMS - Semantifying Content Management Systems. In *ISWC 2011*, 2011.
19. T. Nguyen and M.-Y. Kan. Keyphrase Extraction in Scientific Publications. pages 317–326. 2007.
20. L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *CONLL*, pages 147–155, 2009.
21. N. Seco, T. Veale, and J. Hayes. An intrinsic information content metric for semantic similarity in WordNet. *Proc. of ECAI*, 4:1089–1090, 2004.
22. S. Shekarpour, S. Auer, A.-C. Ngonga Ngomo, D. Gerber, S. Hellmann, and C. Stadler. Keyword-driven sparql query generation leveraging background knowledge. In *International Conference on Web Intelligence*, 2011.
23. Y. Yan, N. Okazaki, Y. Matsuo, Z. Yang, and M. Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *ACL*, ACL '09, pages 1021–1029, 2009.
24. G. Zhou and J. Su. Named entity recognition using an HMM-based chunk tagger. In *ACL '02*, pages 473–480, 2002.