Bootstrapping the Linked Data Web

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, an iterative bootstrapping strategy for extracting RDF from unstructured data. The idea behind BOA is to use the Data Web as background knowledge for the extraction of natural language patterns that represent predicates found on 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. We evaluate our approach on two 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 repository of natural language representations of predicates found on the Data Web.

1 Introduction

While the document-oriented Web aimed at providing information targeted towards humans, the Linked Data Web (short: Data Web) aims to provide knowledge in both human- and machine-readable form. Several approaches have been developed to populate the Data Web. Most of these approaches rely on transforming semi-structured and structured data available on the Web into RDF. The results of the utilization of these approaches can be seen in the significant growth of the Linked Data Cloud from from 12 knowledge bases to 203 knowledge bases in less than four years [9]. While these approaches provide a viable mean to expose semi-structured and structured data on the Data Web, they suffer of one fatal drawback: They can only be applied to 15-20% [3, 8] of the information on the Web, as the rest of the information in the document-oriented Web is only available in unstructured form.

In this paper, we present the an approach that can bootstrap the knowledge available on the Data Web by harvesting triples from unstructured data. Our approach, dubbed BOA[1] (Bootstrapping the Web of Data), starts with the triples available on the Data Web. Then, it extracts natural language patterns that express the predicates found in the triples already available on the Data Web.

[1] http://boa.aksw.org

By using a combination of these patterns and Named Entity Recognition, our approach can identify the labels of instances that stand in the relation expressed by any given predicate. The resulting novel instance knowledge can be finally fed back into the Data Web and reused for extracting even more patterns and triples as well as correcting existing knowledge. 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, it can be used to extract natural language representations of predicates from virtually any language if provided with a Named Entity Recognition service.

Our main contributions are:

1. We present the an approach to bootstrapping the Data Web. Our approach uses knowledge from the Data Web to extract even more knowledge that can be inserted directly into the Data Web.
2. We provide the first knowledge base of natural language representations of predicates found on the Data Web (especially in DBpedia).
3. We present an evaluation of the quality of the natural language patterns extracted automatically by our approach and show that we can extract knowledge from text with a precision of up to 99%.

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

The extraction of entities from natural language (NL) text has been the focus of Information Extraction for a considerable amount of time. Consequently, a multitude of algorithms and tools have been developed for this purpose over the last decades. 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 [12, 11]. Named Entity Recognition (NER) approaches try to discover instances of predefined classes of entities [16, 7]. Finally, Relation Extraction (RE) approaches are used to discover the relations between the entities detected by using NER [13, 17]. 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. [13] 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 [15, 19, 5, 6]. 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 [10, 1]. Services such as Alchemy[2], OpenCalais[3], Extractiv[4] and FOX[5], allow to extract structured information from text. Yet, they mostly rely on classical NLP algorithms in combination with URI lookup on the Data Web. Thus, they are also unfit to harvest RDF from the Web at large scale due to restrictions with respect to coverage.

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[6] [4] is to create the never-ending language learner NELL that can read webpages. To achieve this goal, NELL is fed with the ClueWeb09[7] 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 [14] 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 that uses the Data Web as background knowledge for the large-scale extraction of RDF from natural language, therewith making this knowledge effortlessly integrable into the Data Web. ReadTheWeb and PROSPERA rely on a their own ontology for this purpose. Thus, their results cannot be linked directly into the Data Web. [13] does not generate RDF. In addition, our experiments show that our approach can extract a large number of statements (like PROSPERA and [13]) with a high precision (like ReadTheWeb). For example, 98% of the 2657 statements extracted by BOA on organizations in one iteration were not available in the underlying data source. We minimize the effect of semantic drift by adopting a conservative approach with respect to the patterns that are used to generate RDF from text.

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

3 Approach

In this section we present the BOA framework for extracting natural language representations of relations found on the Data Web. Our workflow is implemented as outlined in Figure 1. We first gather data from the internet by using the corpus extraction module. Alternatively, existing cleaned corpora can be loaded into BOA's corpus repository. Given a set of predicates whose representations are to be learned, the instance knowledge available on the Data Web is harvested. Then, for each predicate, our pattern extraction module searches for prototypical natural language patterns that are specific for this predicate. The patterns are subsequently filtered, scored and finally used for extracting RDF statements out of natural language text. These statements can then be written back into the input knowledge base and the process can be started anew. In the following, we present each of the modules of BOA in more detail. In addition, we exemplify their use and their output by using the example of learning patterns from Wikipedia by using DBpedia.

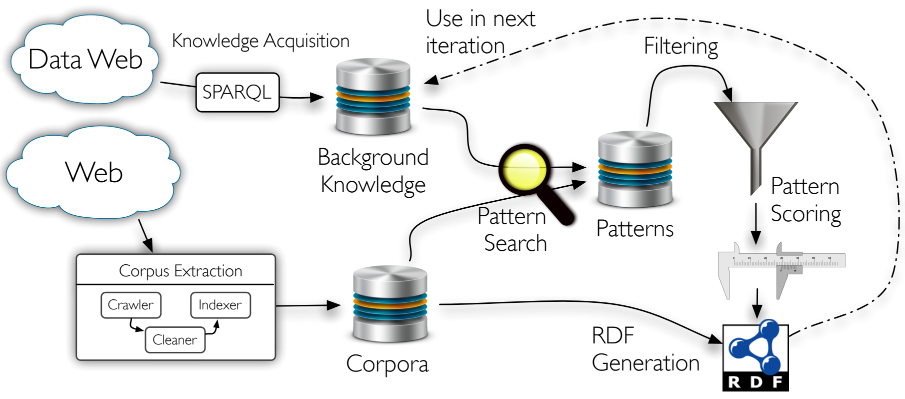


Fig. 1. Overview of the BOA approach.

3.1 Corpus Extraction

The corpus extraction component of BOA consists of three main modules: A crawler, a cleaner and an indexer. The role of the *crawler module* is to retrieve raw text data from the document-oriented Web. The seed pages for this process are determined by querying the Web with the labels of instances that are linked by the predicates whose natural language representations are to be learned. Once presented with a set of seed pages, our crawler can be configured to follow the links in the seed pages up to a certain depth and until a given corpus size is reached. Note that the crawler gets rid of the markup contained in the webpage text. In addition, it allows to integrate corpus-specific preprocessing components so as to enable it to extract raw text. For example, when extracting a corpus from a Wikipedia dump, we used the tool presented in [18] to transform the Wikipedia data from XML to UTF-8 encoded text. The raw text extracted from all pages is merged to a corpus that is sent to the cleaner module.

The *cleaner module* implements the functionality that is necessary to remove noise from the text retrieved by the crawler. It begins by splitting its input into sentences by using the Sentence Boundary Disambiguation provided by the stanford NLP core toolkit[8]. Subsequently, all sentences go through a data cleaning process with 24 UTF-8 compatible filters, introduced in [2]. For example, sentences with two many spaces in relation to length, with uncommon symbols like | [ ] « » and sentences with more than eight capital words in a row are discarded. The cleaned corpus is finally sent to the indexer module.

The *indexer module* allows for the time-efficient search of instance labels and of patterns in our corpus. In BOA, this functionality is implemented by the Lucene indexer[9], which we configured by using the defaults provided by the engine. The corpus extraction process for Wikipedia led to a corpus of 7.6GB that consisted of 44.7 million sentences.

3.2 Knowledge Acquisition

Due to the mere size of the Data Web, extracting natural language representations for all relations found on it would require substantial hardware resources. While our approach is generic enough to be deployed on any knowledge base and on any predicate found on the Data Web, its current implementation demands the input of

- a class C that serves as the `rdfs:domain` or as the `rdfs:range` of the predicates whose representations are to be learned and of
- a knowledge base that serves as background knowledge.

Once C is given, we retrieve all statements that have entities of `rdf:type` C as their subject or objects. By these means, we retrieve all predicates that link such entities to other and ensure that we only retrieve predicates that have been instantiated in the knowledge base of interest. This set of instances is the background knowledge upon which we deploy our pattern extraction approach. Using our knowledge acquisition framework on DBpedia with the classes `:Place`, `:Person` resp. `:Organisation` led to the acquisition of 157, 176 resp. 154 different predicates which were used in 1 to 211306, 260451 resp. 21241 triples, with an average of 2549, 4036 resp. 1313 triples per predicate.

3.3 Pattern Search

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 $\mathfrak{P}$

[8] `http://nlp.stanford.edu/software/tokenizer.shtml`

[9] `http://lucene.apache.org/java/docs/index.html`

is the set of all predicates. In addition, Let $\mathcal{K}$ be the knowledge base that is used as background knowledge. 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 $\texttt{p}$ is the set of pairs $\mathcal{I}(\texttt{p}) = \{(s, o) : (s \text{ p } o) \in \mathcal{K}\}$ that instantiate $\texttt{p}$. In the following, we use $\lambda(x)$ to signify the label of any resource x and $\mu(x)$ to signify x's URI. The pattern search process begins with the even distribution of the set $\mathcal{I}(\texttt{p})$ across pattern search threads. Each of these threads then retrieves all sentences which contain the pairs of labels $(\lambda(s), \lambda(o))$ assigned to it from the input corpus. An example of such sentences for the DBpedia relation `:subsidiary` is shown in Listing 1. If a thread finds a sentence σ that contains both $\lambda(s)$ and $\lambda(o)$, it deletes all tokens that are not found between $\lambda(s)$ and $\lambda(o)$ in σ. The labels are then replaced with the placeholders ?D? for $\lambda(s)$ and ?R? for $\lambda(o)$. We call the resulting string a *natural language representation* of $\texttt{p}$ 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 $\mathcal{P} = (\mu(p), \theta)$, where $\mu(p)$ is p's URI and θ is a natural language representation of p.*

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

```
http://dbpedia.org/resource/Google
  http://dbpedia.org/ontology/subsidiary
  http://dbpedia.org/resource/YouTube .
http://dbpedia.org/resource/Google rdfs:label ''Google''@en .
http://dbpedia.org/resource/YouTube rdfs:label ''Youtube''@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 the Wikipedia corpus for sentences which contain both entity labels returns the sentences depicted in Table 1 amongst others. We can replace "Google" with ?D?, because it is the subject of the `:subsidiary` triple, as well as replace "Youtube" with ?R? because it is the object of the same triple. These substitutions lead to the BOA patterns (`:subsidiary`, "?D?*'s acquisition of* ?R?") and (`:subsidiary`, "?R?*, a division of* ?D?"). For the sake of brevity and in the case of unambiguity, we also call θ "pattern".

Sentence with $\lambda(s)$ before $\lambda(o)$	Sentence with $\lambda(o)$ before $\lambda(s)$
"**Google***'s acquisition of* **Youtube** comes as online video is really starting to hit its stride."	"**Youtube***, a division of* **Google**, is exploring a new way to get more high-quality clips on its site: financing amateur video creators."

Table 1. Example sentences for pattern search.

The search for BOA patterns is completed with a large number of duplicates and requires some post-processing. In addition to the storage of the patterns $\mathcal{M}(\mathtt{p})$ for each $\mathtt{p}$, the post-processing includes the computation of 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.

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.

3.4 Pattern Scoring

The pattern scoring process is carried out in parallel and consists of two steps: selection and score computation. The aim of the *selection* step is to retrieve patterns that abide by a set of conditions that make them fit for RDF extraction. We begin by dismissing patterns which are too long or short (we only consider patterns between three and ten tokens) and patterns which only consist of stop words (we used a list of 41 stop words including "(", "," etc.). In addition we discard all patterns starting with "and" or ", and" since they denote the conjunction of sentences, which leads to ambiguous references when no co-reference analysis or phrase structure analysis is applied. This can be easily seen in the following example: "Davies ‘ son John played first-class cricket for **Tasmania** *and was thrice Mayor of* **Hobart** ." This sentence would lead to "*?D? and was thrice Mayor of ?R?*" being a pattern for the `:capital` predicate, which is clearly wrong. Note that we did not apply any co-reference or phrase structure analysis techniques for performance reasons. The last filter we applied removes all patterns which appear less than three times between labels of the same pair of entities. Note that the statistics used for the pattern scoring step encompass all patterns. The scores are yet only computed for those patterns that abide by the restrictions specified above.

The second part is the actual *score calculation*. The score function integrated in BOA relies on the following set of observations:

1. A good pattern θ for $\mathtt{p}$ is used across several elements of $\mathcal{I}(\mathtt{p})$. This characteristic is modeled by computing the *support* of the pattern.
2. A good pattern θ for $\mathtt{p}$ allows to map ?D? (resp. ?R?) to entities whose `rdf:type` is the `rdfs:domain` (resp. `rdfs:range`) of $\mathtt{p}$. We call this characteristic *typicity.*
3. A good pattern θ is used exclusively to express $\mathtt{p}$, i.e, it occurs in a small number of pattern mappings. We call this last characteristic *specificity.*

We first express these three characteristics of a good pattern formally. Subsequently, we derive our formula for the score of a pattern.

Support We calculate the support $s(\theta, \mathtt{p})$ of the pattern θ for the predicate $\mathtt{p}$ as follows:

$$s(\theta, \mathtt{p}) = \log\left(\max_{(s,o)\in\mathcal{I}(\mathtt{p})} l(s, o, \theta, \mathtt{p})\right)\log(|\mathcal{I}(\mathtt{p}, \theta)|). \tag{1}$$

Since both components of the support are Poisson-distributed (see Figure 2), we use the logarithm to reduce the boosting of very popular patterns.

Typicity A pattern θ is considered to display a high typicity with respect to a predicate $\mathtt{p}$ if it connects only entity labels with match the range and domain restrictions of $\mathtt{p}$. Let d resp. r be functions that map each $\mathtt{p}$ to the highest superclass (except *owl:Thing*) of its `rdfs:domain` resp. `rdfs:range` in the provided ontology. Furthermore, let $\delta(\theta, \sigma)$ resp. $\rho(\theta, \sigma)$ 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 $\zeta(x, y)$ be a function that returns 1 if $x = y$ and else 0. We define the typicity of θ as

$$t(\theta, \mathtt{p}) = \sum_{\sigma\in S}\left(\frac{\zeta(d(\mathtt{p}), \delta(\theta, \sigma)) + \zeta(r(\mathtt{p}), \rho(\theta, \sigma))}{2|S|}\right)\cdot\log(|S| + 1), \tag{2}$$

where $\mathtt{S}$ is the set of sentences used to evaluate the typicity of θ. Note that the first term of the typicity is simply the precision of the pattern. We multiply this factor with the logarithm of (|S|+1) 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 $\delta(\theta, \sigma)$ resp. $\rho(\theta, \sigma)$ is a demanding task, which we solved so far by using a trained NER tagger.

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 $\mathtt{p}$. 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:

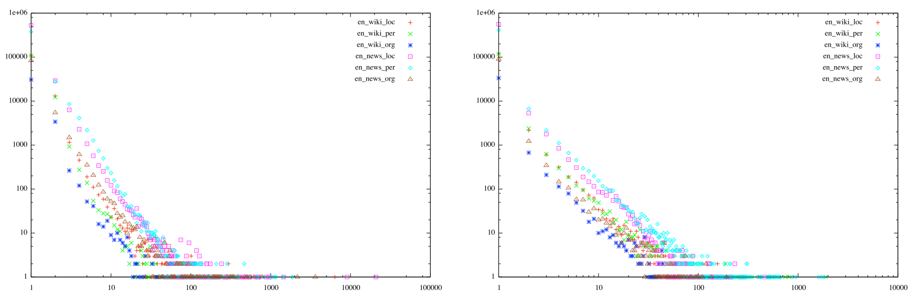
$$i(\theta) = \log\left(\frac{|\mathfrak{P}|}{|\mathfrak{M}(\theta)|}\right), \tag{3}$$

where $\mathfrak{P}$ is the set of all predicates. All three equations can now be combined to the global score $c(\theta, \mathtt{p})$ used by BOA as shown in Equation 4:

$$c(\theta, \mathtt{p}) = s(\theta, \mathtt{p})t(\theta, \mathtt{p})i(\theta). \tag{4}$$

We define the normed score $c_n(\theta, \mathtt{p})$ as the score divided by the local maximum over all patterns belonging to the same pattern mapping to normalize the scores to the interval $[0, 1]$:

$$c_n(\theta, \mathtt{p}) = \frac{c(\theta)}{\max\limits_{\theta'\in\mathcal{M}(\mathtt{p})} c(\theta')}. \tag{5}$$



(a) Distribution of $\max\limits_{(s,o)\in\mathcal{I}(\mathtt{p})} l(s,o,\theta,\mathtt{p})$ (b) Distribution of $|\mathcal{I}(\mathtt{p},\theta)|$

Fig. 2. Distribution of parameters used to compute the *support* of patterns in log-log scale. The y-axis shows the number of patterns

3.5 RDF Generation

The RDF generation is a very delicate process as each iteration generates the input for the subsequent RDF generation. In previous work, semantic drift has been shown to be one of the key problems of this process [4, 14]. In order to maintain a high precision and to avoid semantic drift within the BOA framework, we solely select the top-n patterns θ for each predicate $\mathtt{p}$ according to $c_n(\theta, \mathtt{p})$ for generating RDF. In addition, we filter out those patterns θ which display a normed score below a given threshold as well as a $f((\mu(\mathtt{p}), \theta))$ below a second threshold. 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.

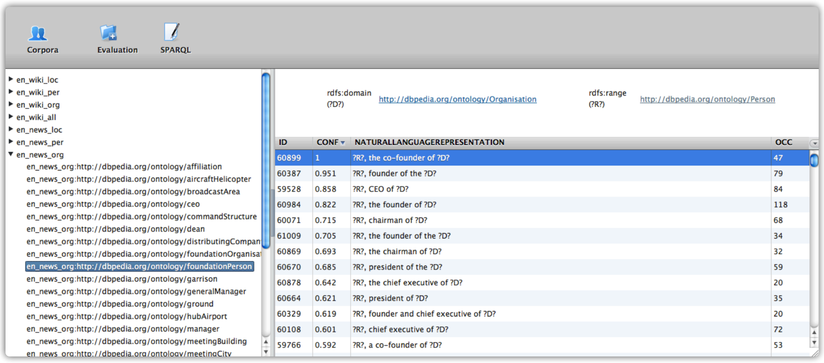
The RDF generation *per se* is carried out as follows: For each pattern θ and each predicate $\mathtt{p}$, we first use the Lucene index to retrieve sentences that contain θ stripped from the placeholders "?D?" and "?R?". These sentences are subsequently processed by a NER tool that is able to detect entities that are of the `rdfs:domain` and `rdfs:range` of $\mathtt{p}$. Thereafter, the first named entities on the left and right of θ which abide by the domain and range restrictions of $\mathtt{p}$ are selected as labels for subject and object of $\mathtt{p}$. Each of the extracted labels is then fed into an URI retrieval service that aims to retrieve the entity e in $\mathcal{K}$ whose label is most similar to the label extracted by BOA. If such an e is found, then we use $\lambda(e)$ in $\mathcal{K}$ as URI for the label detected by BOA. Else, we create a new BOA URI.

Once we have computed URIs, we are finally able to generate RDF triples. The labels retrieved by our URI retrieval approach 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 $\mathtt{p}$. An excerpt of novel statements (with respect to DBpedia) extracted automatically by BOA using Wikipedia and DBpedia can be found in Listing 2. The results of our extraction can be explored via the dashboard shown in Figure 3.

```
http://dbpedia.org/resource/Abdullah_Ahmad_Badawi
    rdfs:label  ‘‘Abdullah Ahmad Badawi’’@en ;
    rdf:type  http://dbpedia.org/ontology/Person .

http://dbpedia.org/resource/Malaysia
    rdfs:label  ‘‘Malaysia’’@en ;
    rdf:type  http://dbpedia.org/ontology/PopulatedPlace ;
    http://dbpedia.org/ontology/leaderName  http://dbpedia.org/
        resource/Abdullah_Ahmad_Badawi .
```

Listing 2. RDF snippet generated by BOA



- en_wiki_loc
- en_wiki_per
- en_wiki_org
- en_wiki_all
- en_news_loc
- en_news_per
- en_news_org
  - en_news_org:http://dbpedia.org/ontology/affiliation
  - en_news_org:http://dbpedia.org/ontology/aircraftHelicopter
  - en_news_org:http://dbpedia.org/ontology/broadcastArea
  - en_news_org:http://dbpedia.org/ontology/ceo
  - en_news_org:http://dbpedia.org/ontology/commandStructure
  - en_news_org:http://dbpedia.org/ontology/dean
  - en_news_org:http://dbpedia.org/ontology/distributingCompan
  - en_news_org:http://dbpedia.org/ontology/foundationOrganisat
  - en_news_org:http://dbpedia.org/ontology/foundationPerson
  - en_news_org:http://dbpedia.org/ontology/garrison
  - en_news_org:http://dbpedia.org/ontology/generalManager
  - en_news_org:http://dbpedia.org/ontology/ground
  - en_news_org:http://dbpedia.org/ontology/hubAirport
  - en_news_org:http://dbpedia.org/ontology/manager
  - en_news_org:http://dbpedia.org/ontology/meetingBuilding
  - en_news_org:http://dbpedia.org/ontology/meetingCity

rdfs:domain (?D?) http://dbpedia.org/ontology/Organisation
rdfs:range (?R?) http://dbpedia.org/ontology/Person

ID	CONF	NATURALLANGUAGEREPRESENTATION	OCC
60899	1	?R?, the co-founder of ?D?	47
60387	0.951	?R?, founder of the ?D?	79
59528	0.858	?R?, CEO of ?D?	84
60984	0.822	?R?, the founder of ?D?	118
60071	0.715	?R?, chairman of ?D?	68
61009	0.705	?R?, the founder of the ?D?	34
60869	0.693	?R?, the chairman of ?D?	32
60670	0.685	?R?, president of the ?D?	59
60878	0.642	?R?, the chief executive of ?D?	20
60664	0.621	?R?, president of ?D?	35
60329	0.619	?R?, founder and chief executive of ?D?	20
60108	0.601	?R?, chief executive of ?D?	72
59766	0.592	?R?, a co-founder of ?D?	53

Fig. 3. Screenshot of the BOA frontend

4 Evaluation

Our evaluation was driven by two main questions:

– Q_1: *Can we use knowledge found on the Data Web to bootstrap the Data Web, i.e., can we find knowledge not yet available on the Data Web?*
– Q_2: *Can we retrieve this knowledge with a high precision, i.e, does the score calculation retrieve the right patterns for each predicate?*

To answer these questions, we evaluated our approach on two different data sets and used DBpedia as source for background knowledge. Note that we only considered the classes `Person`, `Location` and `Organisation` as seed classes for the extraction due to the restrictions of the NER framework we utilized.

4.1 Experimental Setup

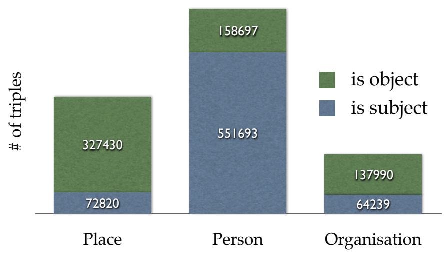
Corpora We used two corpora, which differ in topic, size and writing style. The first corpus, dubbed *en-news*, was described in [2]. It was crawled from news sites in English that were published between the years 2005 and 2010. The corpus contains between 32 million to 50 million unique and cleaned sentences for each year (256.1 million sentences overall). The second corpus, dubbed *en-wiki*, was derived from the English Wikipedia dump of March 2011 without history or discussions entries. The dump was transformed by the tool presented in [18] from XML to UTF-8-encoded text. In contradistinction to *en-news* we did not remove duplicate sentences from *en-wiki* as it did not contain as many duplicates as *en-news*. Overall, the *en-wiki* corpus contains 44.7 million sentences. An overview of the corpora can be found in Table 2.

Background Knowledge We used DBpedia as source for background knowledge. Listing 3 shows an example of the queries used to retrieve properties and instances that are relevant for the classes `Organisation`, `Place` and `Person` from DBpedia (see Figure 4). Overall, the knowledge acquisition process led to 283 different relations ranging from 1 to 471920 triples, with an average of 4639 triples per relation. Note that the evaluation was carried out independently for each of the 6 possible combinations of seed classes and corpora.

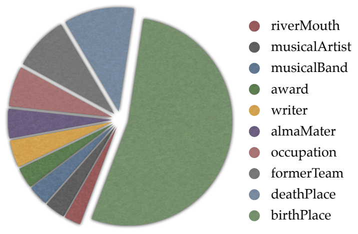
Parameters We limited the number of sentences returned by Lucene for the pattern search to 25000. We also excluded all patterns $\mathcal{P} = (\mu(\mathrm{p}), \theta)$ with $f(\mathcal{P}) < 20$. We used up to 500 sentences to calculate the typicity of the patterns (see Equation 2). For the selection of sentences for the RDF generation, we only used the top-1 and top-2 patterns for each predicate. The evaluation of the accuracy of each pattern was carried out manually by two evaluators on 100 statements that were selected randomly. The inter-annotator agreement was computed by using Cohen's Kappa. Since the precision of the entity extraction algorithm is out of scope of this paper, we considered a triple as correct if the sentence it was generated from contained the right labels for named entities that matched the `rdf:type` of the domain and range of `p`. Furthermore, sentences which contained references to correct entities (e.g., pronouns) but not the entity labels themselves were considered to be false positives.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpo: <http://dbpedia.org/ontology/>
SELECT ?s ?sLabel ?prop ?o ?oLabel ?domain ?range
WHERE {
  ?s rdf:type dbpo:[Organisation|Person|Place] .
  ?s rdfs:label ?sLabel .
  ?o rdfs:label ?oLabel .
  [?o ?prop ?s|?s ?prop ?o] .
  FILTER (lang(?sLabel) = en && lang(?oLabel) = en) .
  ?prop rdfs:range  ?range .
  ?prop rdfs:domain ?domain .
}
```

Listing 3. SPARQL query template used for knowledge acquisition



(a) Distribution of triples used for pattern search



(b) Distribution of triples for 10 most frequent predicates

Fig. 4. Overview of the knowledge extraction on DBpedia

	wiki	**news**
Language	English	English
Topic	General knowledge	Newspaper articles
Number of lines	44.7	256.1
Number of words	1,032.1	5,068.7
Number of characters	5,689	30,289.7
Number of unique words	5.9	26.3

Table 2. Corpora statistics. (All figures in millions.)

4.2 Results and Discussion

The results of our evaluation are shown in Table 3. We reached an average inter-annotator agreement of 0.9175. Our approach performed consistently best on predicates related to organizations and worst on those related to locations. When using the *en-wiki* as corpus, our worst precision was 90.5% on locations when using the top pattern. This value improved to 93% when using the top-2 patterns. Our overall best precision of 99% clearly answers question Q_2 positively. Our good performance on *en-wiki* is due to the encyclopedic character of this dataset. Overall, *en-wiki* contained a (in relation to its size) relatively larger number of sentences that expressed the relations found in DBpedia. Consequently, we could rely on a relatively larger number of good exemplary sentences and compute more accurate scores.

The *en-news* corpus was less reliable in this respect. It contained less information (and consequently more noise) than *en-wiki* and thus led to worse statistics and patterns. This characteristic of the data set was especially noticeable for predicates of the `Location` class, as locations are very generic and can thus occur in a large number of contexts that do not express the predicate `p` used as input. Consequently, we solely reached precisions between 57% and 61.5% for `Location` on the *en-news* corpus. On the class `Organisation`, we still reached precisions of up to 93%.

	Top 1 Patterns						Top 2 Patterns					
	en-wiki			**en-news**			**en-wiki**			**en-news**		
	LOC	PER	ORG	LOC	PER	ORG	LOC	PER	ORG	LOC	PER	ORG
Rater 1	88%	97%	99%	61%	73%	91%	94%	96%	96%	57%	67%	92%
Rater 2	93%	97%	99%	62%	74%	91%	92%	96%	95%	57%	68%	94%
Intersection	84%	96%	99%	59%	73%	91%	94%	96%	95%	57%	67%	92%
Average	90.5%	97%	99%	61.5%	73.5%	91%	93%	96%	95.5%	57%	67.5%	93%
κ	0.68	0.66	1	0.94	0.97	1	0.9	1	0.88	1	0.98	1

Table 3. Evaluation results for top-1 and top-2 patterns.

Note that the patterns extracted from *en-wiki* are by no means bound to *en-wiki* for the extraction of triples. Thus, we can apply the patterns retrieved using *en-wiki* on any corpus. Consequently, the precision scores achieved on *en-wiki* reflect best the overall capabilities of our extraction approach. For example, when applying the top *en-wiki* pattern for predicates from each of the three classes `Person`, `Location` and `Organisation` with the Google index and only considering the top-20 pages returned, we achieved a precision of 95% of the `:spouse`, 95% of the `:capital` and 100% on the `:subsidiary` predicates. Examples of natural language representations extracted by BOA are shown in Table 5.

We measured how much new correct knowledge we were able to generate by counting the number of statements that we generate that could not be found in DBpedia and multiplying it with the precision of our approach. Table 4 shows that we extracted more that 13,000 new correct facts in one iteration, therewith answering also Q_1 with a clear "yes". Examples of these new facts are shown in Table 6. As our evaluation shows, our approach is particularly well suited for extracting knowledge on organizations. For example, 98% of the facts considered in our evaluation and extracted by using news data could not be found in DBpedia. When using Wikipedia as text data, 99% of the statements that were evaluated on the organization data were not available in DBpedia. When assuming an even distribution of correct facts and of their inclusion in DBpedia, this implies that 2494 of the 2567 statements on organizations extracted from Wikipedia by using DBpedia are not to be found in DBpedia. Note that one iteration on all predicates linked to organizations on the *en-news* corpus lasts about 15 minutes, therewith showing that our approach is fully suitable for the large-scale extraction of new knowledge from the Web.

A direct comparison of our approach with those presented in [4, 14, 13] cannot be carried out due to the fact that we operate on different background knowledge and on different text corpora. Nevertheless, our evaluation on *en-wiki* shows that although we do not use a reasoner, we extract patterns with a precision equal or superior to those extracted by PROSPERA [14] with a reasoner. The precision achieved by PROSPERA without a reasoner lies significantly below that of BOA. The same holds for the approach presented in [13]. In addition, we extract significantly more correct statements in our first iteration than any of

	en-wiki			*en-news*		
	LOC	PER	ORG	LOC	PER	ORG
Triples extracted/in DBpedia	1465/138	8817/183	2567/48	488/52	903/44	916/7
Evaluated triples/in DBpedia	100/8	100/1	100/1	100/1	100/7	100/0
Precision (average)	90.5%	97%	99%	61.5%	73.5%	91%
New true statements	1200	8375	2494	268	631	827
Pattern mappings/patterns	62/1045	72/612	59/241	49/3832	70/7294	55/1077

Table 4. Overview of extraction statistics of first iteration

the previous approaches even when using a corpus that is more than 650 times smaller than ClueWeb09, therewith hinting towards a higher recall.

Relation	Top 2 Pattern	
URI Domain/Range	**en-wiki**	**en-news**
birthPlace Person/PopulatedPlace	1. *D* was born in *R* 2. — (*D* , the mayor of *R*)	1. *D* has been named in the *R* 2. *D*, MP for *R*
foundationPerson Organisation/Person	1. *R* , co-founder of *D* 2. *R* , founder of *D*	1. *R*, the co-founder of *D* 2. *R*, founder of the *D*
subsidiary Organisation/Organisation	1. *R* , a subsidiary of *D* 2. — (*R* , a division of *D*)	1. *R*, a division of *D* 2. *D*'s acquisition of *R*
riverMouth River/BodyOfWater	1. *D* , which flows to *R* 2. *D* , a tributary of the *R*	1. — (*D* empties into the *R*) 2. — (*D*, which joins the *R*)
leaderName PopulatedPlace/Person	1. *D* 's Prime Minister *R* 2. *R* , the Prime Minister of *D*	1. *D*'s Prime Minister *R* 2. *D* for talks with President *R*
capital PopulatedPlace/City	1. *R* , the capital of *D* 2. *R* , capital of *D*	1. *R*, the capital of *D* 2. *R*, capital of *D*

Table 5. Top-2 natural language representations for six most used relations in evaluation. "—" means that no natural language representation was found, patterns in brackets are next in line but were not used for the evaluation because they did not fulfill the threshold requirements.

5 Conclusion and Future Work

In this paper, we presented BOA, an approach for bootstrapping the Data Web. Our approach is based on using instance data for predicates found on the Data Web and retrieving natural language representations for these predicates. Based on these representations, we can extract sentences from natural language text that contain both named entities and patterns. These sentences can then be used to extract (in particular novel) knowledge from the document-oriented Web and integrating this knowledge into the Data Web. Our evaluation shows that when combining knowledge from DBpedia with text from Wikipedia, we achieve precision scores beyond 90% and can retrieve more than 12,000 facts that (13,000 with *en-news*) were not previously found in DBpedia within one iteration. In future work, we will aim to deploy our approach on large data sets such as ClueWeb09 to discover even more facts. Note that our approach can thus be used on most languages whose grammar adheres roughly to the SPO idea. The potential of the approach presented herein is immense, as it could promote the Data Web to the lingua franca for a large number of applications including machine translation, named entity recognition and speech generation.

wiki-loc		
Westlake High School	`tenant`	Chaparral Stadium
Boston College	`tenant`	Higgins Hall
Konzerthaus Berlin	`architect`	Karl Friedrich Schinkel
Villa Foscari	`architect`	Andrea Palladio
wiki-per		
Elvin Jones	`birthPlace`	Pontiac , Michigan
Ernest Reyer	`birthPlace`	Marseilles
Henri Curiel	`deathPlace`	Paris
Carrero Blanco	`deathPlace`	Madrid
wiki-org		
Time Warner	`subsidiary`	DC Comics
Interscope Records	`subsidiary`	Star Trak Entertainment
Heavy Brigade	`notableCommander`	James Yorke Scarlett
Federal Department of the West	`notableCommander`	John C. Fremont
news-loc		
Badakhshan	`capital`	Faizabad
Quetta	`capital`	Baluchistan
Bulgaria	`leaderName`	Boyko Borisov
Japan	`leaderName`	Taro Aso
news-per		
Leyla Rodriguez Stahl	`spouse`	Abel Pacheco
Sehba Musharraf	`spouse`	Pervez Musharraf
Kelly Osbourne	`father`	Ozzy Osbourne
Svetlana Alliluyeva	`father`	Josef Stalin
news-org		
College of Cardinals	`dean`	Cardinal Joseph Ratzinger
Yale University School of Architecture	`dean`	Robert A.M. Stern
Aldi	`foundationPerson`	Theo Albrecht
World Wrestling Entertainment	`foundationPerson`	Vince McMahon

Table 6. Triples extracted from evaluation data set not present in DBpedia.

References

1. Benjamin Adrian, Jörn Hees, Ivan Herman, Michael Sintek, and Andreas Dengel. Epiphany: Adaptable rdfa generation linking the web of documents to the web of data. In *EKAW*, pages 178–192, 2010.

2. C. Biemann, G. Heyer, U. Quasthoff, and M. Richter. The Leipzig Corpora Collection – Monolingual Corpora of Standard Size. In *Proceedings of the 4th Conference on Corpus Linguistics (CL)*, 2007.
3. R Blumberg and Shaku Atre. The problem with unstructured data. *DM Review*, 13(February 2003):42–49, 2003.
4. Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
5. James R. Curran and Stephen 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. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL*, ACL '05, pages 363–370, 2005.
7. Jenny Rose Finkel and Christopher D. Manning. Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data. In *ACL '10*, pages 720–728, 2010.
8. A. Gaag, A. Kohn, and U. Lindemann. Function-based solution retrieval and semantic search in mechanical engineering. In *IDEC '09*, pages 147–158, 2009.
9. Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space.* 2011.
10. David Huynh, Stefano Mazzocchi, and David R. Karger. Piggy bank: Experience the semantic web inside your web browser. In *ISWC*, pages 413–430, 2005.
11. Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *SemEval '10*, pages 21–26, 2010.
12. 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.
13. M Mintz, S Bills, R Snow, and D Jurafsky. Distant supervision for relation extraction without labeled data. *ACL*, pages 1003–1011, 2009.
14. Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *WSDM*, pages 227–236, Hong Kong, 2011.
15. Thuy Nguyen and Min-Yen Kan. Keyphrase Extraction in Scientific Publications. pages 317–326. 2007.
16. Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *CONLL*, pages 147–155, 2009.
17. Yulan Yan, Naoaki Okazaki, Yutaka Matsuo, Zhenglu Yang, and Mitsuru Ishizuka. Unsupervised relation extraction by mining wikipedia texts using information from the web. In *ACL*, ACL '09, pages 1021–1029, 2009.
18. Hasebe Yoichiro. Method for using wikipedia as japanese corpus. *Doshisha studies in language and culture*, 9(2):373–403, 2006-12.
19. GuoDong Zhou and Jian Su. Named entity recognition using an HMM-based chunk tagger. In *ACL '02*, pages 473–480, 2002.