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 then 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.

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 [1], they rely on background data that encompasses solely 15-20% [5] 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. For example, while the Wikipedia text fragment "`... reputedly designed by Robert Mills, architect of the Washington Monument ...`" states that the triple "`dbr:Washington_Monument dbo:architect dbr:Robert_Mills`" holds, this triple is not included in DBpedia 3.7. In addition, being able to convert natural language to structured data makes manifold applications such as using SPARQL for question answering [14] possible by allowing that the pattern `born in` from questions such as `Which actors were born in Germany?` to be mapped explicitly to the relation `dbo:birthPlace`.

A. ten Teije et al. (Eds.): EKAW 2012, LNAI 7603, pp. 87–96, 2012.

In this paper, we extend the BOA framework[1] presented in [6]. The goal of the BOA framework is to allow extracting structured data as RDF from unstructured data. Unlike many approaches (e.g., [2]) which start with their own ontologies and background knowledge as seeds, BOA makes use of the Linked Data Web to retrieve high-confidence natural-language patterns that express the predicates available in the Data Web. In contrast to its previous model, BOA uses a supervised machine learning approach trained on a set of manually annotated patterns to recognize high-confidence 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, our extension of 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.

The main contributions of this paper are as follows: (1) We present the novel approach implemented by the BOA framework and apply it to corpora written in English and in German. (2) We provide a multilingual library of natural-language representations of predicates found on the Data Web (especially in DBpedia). (3) We present a set of features that can be used to distinguish high-quality from poor natural-language patterns for Data Web predicates. (4) 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 sum up 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) [8], Named Entity Recognition (NER) [4] and Relation Extraction (RE) [10]. 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. [10] coined the term "distant supervision" to describe this paradigm but developed an approach that led to extractors with a low precision (approx. 67.6%). Services

[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`

such as Alchemy[2], FOX [12] and Spotlight [9] reach better precision scores and 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[3] [2] is to create the never-ending language learner NELL that can read webpages. To achieve this goal, NELL is fed with the ClueWeb09[4] data set and an initial ontology. 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 [11] is similar to that of NELL but relies on the iterative harvesting of n-grams-itemset patterns that allow generalizing NL patterns found in text.

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 their own ontology for this purpose. Moreover, BOA can generate 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 [10]) with a high precision (like ReadTheWeb).

3 Approach

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[5]. When provided with 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 [9]. 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.

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

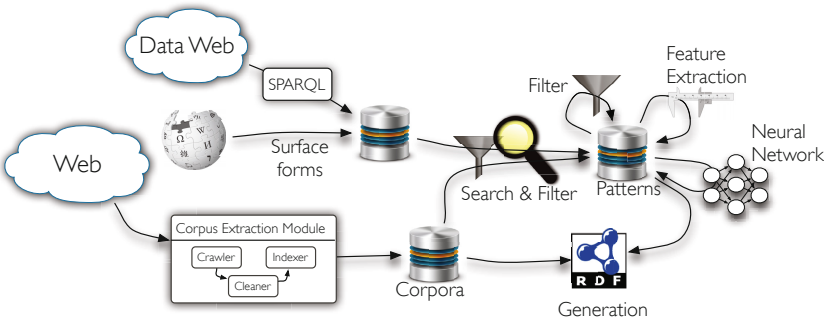

Fig. 1. Overview of the BOA approach

3.1 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 [9]. Overall, the average number of surface forms per resource was 1.66 for German and 2.36 for English. 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 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* $\mathcal{P} = (\mu(\mathtt{p}), \theta)$*, where* $\mu(\mathtt{p})$ *is* $\mathtt{p}$*'s URI and* θ *is a natural-language representation of* $\mathtt{p}$*.*

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

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

Listing 1. RDF snippet used for pattern search

Table 1. Example sentences for pattern search

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** ..."

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 1 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 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 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 compute (1) the total number of occurrences of a BOA pattern $\mathcal{P}$, dubbed $f(\mathcal{P})$; (2) 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 (3) $\mathcal{I}(\mathtt{p}, \theta)$ 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.

3.2 Feature Extraction

The feature extraction is applied on all patterns which overcome both filtering processes. 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. The first three features BOA relies upon are the support, specificity and typicity as described in [6]. In addition, we rely on the three supplementary features dubbed IICM, ReVerb and Tf-Idf. 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 was introduced in [13] and is based on the Jiang-Conrath similarity measure [7]. 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. If no such synsets are found we use the tokens of the *rdfs:label* of $\mathcal{M}(\mathtt{p})$. We then apply the IICM measure pairwise to these tokens and the tokens derived from one $\mathcal{M}(\mathtt{p})$

assigned pattern's NLR. The IICM score for one pattern is then the maximal value of the similarity values of all pairs. **ReVerb** has been introduced in [3] and distinguishes good from bad relation phrases by measuring how well they abide to a predefined part-of-speech-based regular expression. 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 3.1) 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. The **Tf-Idf** features are an adaption of the tf-idf score used in information retrieval and text mining. The intuition behind this feature 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, the 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:

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

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 $\mathtt{p}$.

3.3 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 exist 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.

3.4 RDF Generation

The generation of RDF out of the knowledge acquired by BOA is the final step of the extraction process and 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 `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 [12]. If this service returns a URI, then we use it for the label detected by BOA. Else, we create a new BOA URI. By applying our approach, we were able to extract the triples shown in Listing 2 from the text fragment " `... reputedly designed by Robert Mills, architect of the Washington Monument.`".

```
dbr:Washington_Monument dbo:architect dbr:Robert_Mills .
dbr:Washington_Monument rdf:type dbo:Building .
dbr:Washington_Monument rdfs:label "Washington Monument"@en .
dbr:Robert_Mills rdf:type dbo:Architect .
dbr:Robert_Mills rdfs:label "Robert Mills"@en .
```

Listing 2. RDF snippet generated by BOA

Note that `dbr:Washington_Monument dbo:architect dbr:Robert_Mills` is not included in DBpedia but explicitly stated in Wikipedia [6].

4 Evaluation

The aim of our evaluation was three-fold. First, we aimed at testing how well BOA performs on different languages. To achieve this goal, we applied BOA to German and English. Our second goal was to determine the accuracy 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 not yet distinguish between different time expressions and conjugations. We evaluated our approach on the 4 corpora described in Table 2.

Table 2. Statistical overview of German and English text corpus

Corpus	Sentences	Tokens	Unique Tokens	Tokens per Sentence
en-wiki	58.0M	1,240.6M	6.8M	21.4
en-news	214.3M	4,745.1M	17.6M	22.1
de-wiki	24.6M	428.4M	6.7M	17.4
de-news	112.8M	2,062.1M	18.0M	18.3

[6] The functionality used to extract these triples is publicly available via a REST webservice described at `http://boa.aksw.org`

4.1 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 our 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 whose score was 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 2. It is easy to see that networks trained to achieve an error rate of maximally 5% performed best in our experiments.

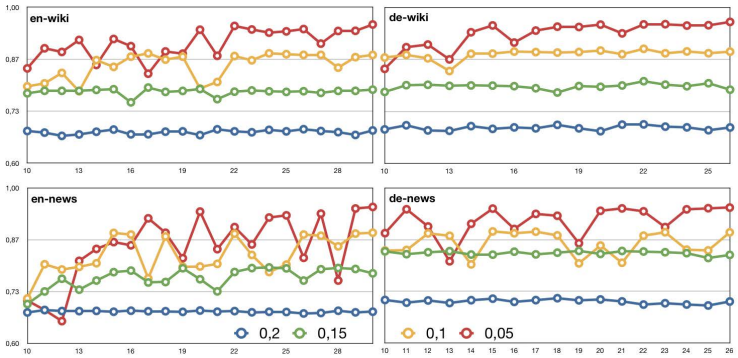

Fig. 2. Learning curves of BOA's neural networks. The x-axis shows the size of the hidden layer while the y-axis shows the accuracy achieved by the neural network. The different colors stand for different maximal error rates.

4.2 Multi-linguality

Enabling BOA to process languages other than English requires solely the alteration of the NER tools and POS taggers. 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 acceptable precision (see subsequent section).

4.3 Accuracy

The results of our experiments on accuracy are shown in Table 3. 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. For the top-100 scored triples we achieved a precision superior to 90% overall on the English data sets. This value is comparable to that achieved by the previous versions of BOA [6]. 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 more than twice the amount of triples. The same holds for the number of patterns and pattern mappings as shown in Table 3.

Table 3. Results of one iteration of the BOA framework

	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	1,829	798	655	42
Number of found triples	80773	3081	10793	925
Precision top-100 triples	92%	70%	91%	74%

An excerpt of the new knowledge extracted by BOA is shown in Listing 3. 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 deal and thus extract this triple.

```
1 Chinese spokenIn Malaysia .
2 Chinese spokenIn  China .
3 Weitnau administrativeDistrict boa:Oberallgäu .
4 Memmingerberg administrativeDistrict boa:Unterallgäu .
5 ESV_Blau-Rot_Bonn ground Bonn .
6 TG_Würzburg ground Würzburg .
7 Intel_Corporation subsidiary McAfee .
8 Iomega subsidiary ExcelStor_Technology .
```

Listing 3. 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 using surface forms to retrieve entities. 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. Auer, S., Lehmann, J., Ngonga Ngomo, A.-C.: Introduction to Linked Data and Its Lifecycle on the Web. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 1–75. Springer, Heidelberg (2011)
2. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: AAAI (2010)
3. Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: EMNLP, pp. 1535–1545. ACL (2011)
4. Finkel, J.R., Manning, C.D.: Hierarchical joint learning: improving joint parsing and named entity recognition with non-jointly labeled data. In: ACL 2010, pp. 720–728 (2010)
5. Gaag, A., Kohn, A., Lindemann, U.: Function-based solution retrieval and semantic search in mechanical engineering. In: IDEC 2009, pp. 147–158 (2009)
6. Gerber, D., Ngonga Ngomo, A.-C.: Bootstrapping the linked data web. In: 1st Workshop on Web Scale Knowledge Extraction ISWC (2011)
7. Jiang, J.J., Conrath, D.W.: Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In: ROCLING X, p. 9008 (September 1997)
8. Kim, S.N., Medelyan, O., Kan, M.-Y., Baldwin, T.: Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: SemEval 2010 (2010)
9. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia Spotlight: Shedding Light on the Web of Documents. In: I-SEMANTICS, pp. 1–8. ACM (2011)
10. Mintz, M., Bills, S., Snow, R., Jurafsky, D.: Distant supervision for relation extraction without labeled data. In: ACL, pp. 1003–1011 (2009)
11. Nakashole, N., Theobald, M., Weikum, G.: Scalable knowledge harvesting with high precision and high recall. In: WSDM, Hong Kong, pp. 227–236 (2011)
12. Ngonga Ngomo, A.-C., Heino, N., Lyko, K., Speck, R., Kaltenböck, M.: SCMS – Semantifying Content Management Systems. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part II. LNCS, vol. 7032, pp. 189–204. Springer, Heidelberg (2011)
13. Seco, N., Veale, T., Hayes, J.: An intrinsic information content metric for semantic similarity in WordNet. In: Proc. of ECAI, vol. 4, pp. 1089–1090 (2004)
14. Unger, C., Bühmann, L., Lehmann, J., Ngonga Ngomo, A.-C., Gerber, D., Cimiano, P.: Sparql template-based question answering. In: Proceedings of WWW (2012)