Improving Efficiency and Accuracy in Multilingual Entity Extraction

Joachim Daiber
Computational Linguistics
Rijksuniversiteit Groningen
Groningen, The Netherlands
daiber.joachim@gmail.com

Max Jakob
Neofonie GmbH
Robert-Koch-Platz 4
10115 Berlin, Germany
max.jakob@neofonie.de

Chris Hokamp
Lang. and Inf. Technologies
University of North Texas
Denton, TX, USA
chris.hokamp@gmail.com

Pablo N. Mendes
Kno.e.sis Center, CSE Dept.
Wright State University
Dayton, OH, USA
pablo@knoesis.org

ABSTRACT

There has recently been an increased interest in named entity recognition and disambiguation systems at major conferences such as WWW, SIGIR, ACL, KDD, etc. However, most work has focused on algorithms and evaluations, leaving little space for implementation details. In this paper, we discuss some implementation and data processing challenges we encountered while developing a new multilingual version of DBpedia Spotlight that is faster, more accurate and easier to configure. We compare our solution to the previous system, considering time performance, space requirements and accuracy in the context of the Dutch and English languages. Additionally, we report results for 9 additional languages among the largest Wikipedias. Finally, we present challenges and experiences to foment the discussion with other developers interested in recognition and disambiguation of entities in natural language text.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*language models, text analysis*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis—*linguistic processing*

General Terms

Algorithms, Performance

Keywords

Named Entity Recognition, Entity Linking, Information Extraction

ISEM'13, September 04 - 06 2013, Graz, Austria

ACM 978-1-4503-1972-0/13/09...$15.00
http://dx.doi.org/10.1145/2506182.2506192.

1. INTRODUCTION

DBpedia Spotlight [3] is an open source project developing a system for automatic annotation of DBpedia entities in natural language text. It provides programmatic interfaces for phrase spotting (recognition of phrases to be annotated) and disambiguation (entity linking) as well as various output formats (XML, JSON, RDF, etc.) in a REST-based web service. The standard disambiguation algorithm is based upon cosine similarities and a modification of TF-IDF weights (using Apache Lucene[1]). The main phrase spotting algorithm is exact string matching, which uses LingPipe's[2] Aho-Corasick implementation.

The project has focused initially on the English language. However, since DBpedia Spotlight's models are learned from Wikipedia, it should be possible to adapt the system to any other language that has a Wikipedia edition. In addition, although fairly intensively used by researchers[3] and others,[4,5] the current implementation can be improved in certain aspects.

In this paper, we describe how we enhanced the performance and accuracy of our entity recognition and disambiguation components and discuss challenges encountered while adapting DBpedia Spotlight to work with other languages. For this experiment, we focused on the Dutch language. As a result, we present a new version of the system, which is faster, more accurate and improves the ease of internationalization. Demonstrations of internationalized versions are provided in English and Dutch and models for 9 additional languages are made available via the supporting material for this submission.

2. IMPLEMENTATION

2.1 Phrase Spotting

Phrase spotting is the task of finding phrases in a text that later should be linked to DBpedia entities. The stan-

[1] `http://lucene.apache.org/`
[2] `http://alias-i.com/lingpipe`
[3] As of the time of writing, Google Scholar reports 108 citations to the main DBpedia Spotlight article.
[4] `http://bit.ly/A6RE27`
[5] `http://git.io/qB3n6g`

dard phrase spotting algorithm used by DBpedia Spotlight is very light on its NLP demands, only requiring one language-dependent step: tokenization. Recognition of spot candidates is then performed by substring matching using a kind of prefix tree and the Aho-Corasick algorithm.

Our goal is to have the best possible phrase spotting quality and time performance with little memory overhead. Previous work [2] showed increased precision when using NLP-informed phrase spotting methods. Therefore, we developed a Spotter that proceeds in two steps.

First step:. In the first step, candidates for possible annotations are generated. There are two implementations of this step. In the language-independent implementation, the candidates are generated by traversing a finite state automaton encoding all possible sequences of tokens that form known spot candidates.

In the language-dependent implementation, candidates are generated using three methods: 1. identifying all sequences of capitalized tokens, 2. identifying all noun phrases, prepositional phrases and multi word units, 3. identifying all named entities. Methods 2 and 3 are performed using Apache OpenNLP[6] models for phrase chunking and Named Entity Recognition.

For English and German, all required OpenNLP models are readily available. For Dutch, we created a chunker model from the Lassy Small corpus, a corpus of Dutch language texts with automatic syntactic annotations which were manually corrected [5]. The resulting phrase chunker detects Noun Phrase chunks (NP), Prepositional Phrase chunks (PP) and Multi Word Units (MWUs). We use this model in addition to the publicly available Dutch Named Entity Recognition models.

Second step:. As our goal is to achieve the best possible precision, the second step selects the best candidates from the set of phrases generated in step 1. First, overlaps in the candidates are resolved based on a score and a preference-based choice for the method that generated them.[7] Second, all candidates that fall below a specified score threshold are removed. We compute the score for each spot candidate as a linear combination of features, since this gives us flexibility in treating certain kinds of phrases differently. One of our features is the annotation probability $P(\mathsf{annotation}|s)$. Given a set of mentions of article links e with anchor text s (spot), we estimate the annotation probability as: $P(\mathsf{annotation}|s) = \sum_e \mathrm{count}(e,s)/\mathrm{count}(s)$. We identified several cases of phrases where this annotation probability is consistently lower than the general case. Acronyms are one example of this phenomenon (e.g. 'AIG', 'IBM'). Hence, we add binary features that are triggered for these kinds of cases to produce the overall score. We determine the weights for the components of this score from held-out data via linear regression. This method provides the additional advantage of automatically estimating the optimal cut-off threshold from the data.

2.2 Disambiguation

Disambiguation in our build is performed using the generative probabilistic model from [1]. The score for an entity e given the phrase s and context c is calculated as a combination of P(e), P($s|e$) and P($c|e$). The combination can be either a linear regression mixture or, in accordance with the generative model, the product of the individual values. Given the Wikipedia data set M consisting of article links with their anchor texts and textual context, we estimate the required probability distributions from the raw counts using the maximum likelihood approach presented in [1]. $P_{\mathrm{LM}}(t)$ is the smoothed unigram language model estimated over all tokens in the data set and weighed by the parameter λ whose value we adopted from [1].

$$P(e) = \frac{\mathrm{count}(e)}{|M|} \tag{1}$$

$$P(s|e) = \frac{\mathrm{count}(e,s)}{\mathrm{count}(e)} \tag{2}$$

$$P(c|e) = P_e(t_1) \cdot P_e(t_2) \cdot P_e(t_3) \cdot ... \cdot P_e(t_n) \tag{3}$$

$$P_e(t) = \lambda P_{e\_\mathrm{ML}}(t) + (1-\lambda) P_{\mathrm{LM}}(t) \tag{4}$$

$$P_{\mathrm{e\_ML}}(t) = \frac{\mathrm{count}_e(t)}{\sum_t \mathrm{count}_e(t)} \tag{5}$$

In this generative model, probabilities are negligibly small and are represented in logarithmic space to avoid underflows. To produce a more useful score for each entity, we normalize the score via the softmax function and obtain a final disambiguation score between 0.0 and 1.0.

For each phrase and its surrounding context tokens, we further generate a `NIL` entity, which represents the hypothesis that the context and phrase were not generated by any known entity. We use the formulas for $P(\mathtt{NIL})$, $P(s|\mathtt{NIL})$ and $P(c|\mathtt{NIL})$ from [1] to calculate the score for the `NIL` entity. All entity candidates with a lower score than the `NIL` entity are removed.

$$P(\mathtt{NIL}) = \frac{1}{|M|} \tag{6}$$

$$P(s|\mathtt{NIL}) = \prod_{t \in S} P_{\mathrm{LM}}(t) \tag{7}$$

$$P(c|\mathtt{NIL}) = \prod_{t \in C} P_{\mathrm{LM}}(t) \tag{8}$$

As above, P_{LM} is the smoothed general language model probability of a token that we estimate over all tokens imported to the system as context of an entity mention.

2.3 Indexing

The indexing process of our model consists of two phases: First, we collect the raw counts necessary for our models from Wikipedia. Some language-dependent pre-processing is performed, including stemming and the removal of stopwords. We use PigNLProc,[8] a collection of Pig Latin scripts and utilities focused on Wikipedia and DBpedia that we adapted for this work. Apache Pig is part of the Hadoop ecosystem, providing a high-level abstraction of MapReduce using an SQL-like syntax. In the second step, the raw counts are serialized into efficient data structures that the system can de-serialize at runtime. This step can be performed on a commodity PC and takes a few minutes for Dutch and 1-2 hours for the English version.

For the annotation probability, we must find count(s), the total number of string occurrences of a phrase in the corpus. Since performing a string search for each possible phrase in the entire corpus is not feasible, we collect all n-grams (where

[6] `http://opennlp.apache.org/`

[7] In the order PER≻ORG≻LOC≻MISC≻NP≻MWU≻PP≻FSA lookup≻Capitalized Sequence.

[8] `https://github.com/dbpedia-spotlight/pignlproc/`

n is a parameter set to 5 by default) in the corpus and in order to obtain the number of times s has occurred, we intersect this set with the set of known phrases. This simplification makes the process feasible but also introduces limitations: phrases with more than n tokens are not counted and phrases with more than one token can contain other phrases as substrings (e.g. "Apple MacBook" and "Apple"), which skews the counts of these substring phrases. In our system, we correct the latter issue by subtracting the counts of phrases from the total counts of their phrase substrings.

2.4 Data Storage, Models and Configuration

When selecting our preferred data storage implementation, we considered the following criteria: compatibility with the Apache 2.0 license, optimization for read operations, type of data serialization, ease of integration and scalability. While some full-fledged DBMS, such as Apache Cassandra[9] met most of our criteria, we decided to use an in-memory model efficiently serialized using Kryo,[10] since this allows maximum retrieval performance, high flexibility, smaller external dependencies and deliberate optimizations based on our knowledge of the data. Optionally, JDBM3[11] can be used for disk-based access on low memory systems. We share our performance evaluations of suitable data storages in the supporting material for this submission.[12] To reduce the complexity of the configuration, we introduced a self-contained model directory structure that is produced by the indexing module and can be run with no further configuration using the DBpedia Spotlight server.

3. EVALUATION

Our aim was to produce a system providing easier and faster indexing, faster and more accurate runtime performance and simple internationalization. The evaluation was carried out on our build as well as the current build of DBpedia Spotlight for Dutch and English. We evaluated indexing and runtime performance, phrase spotting and disambiguation. Additional to English and Dutch, where we had previous systems to compare against, we provide these values for 9 additional languages.

3.1 Performance: Runtime and Footprint

Annotation time. To measure time performance (Table 1), we annotated a small corpus of around 500 randomly selected articles from the Reuters news corpus using both systems with their default settings.[13]

Language	Model	Avg	Total
Dutch	Our build	1.20s	601.39s
	Current	9.52s	4758.31s
English	Our build	1.28s	640.2s
	Current public endpoint	5.72s	2803.22s

Table 1: Performance evaluations

[9] http://cassandra.apache.org/
[10] http://code.google.com/p/kryo/
[11] https://github.com/jankotek/JDBM3
[12] http://git.io/xvzPGw
[13] Except for the current build of the English version marked as *Public endpoint*, all tests were performed on the same personal computer.

Disk and Memory Footprint. In Table 2, we report the disk and memory footprint for our build and for two versions of the current build: in Current (1), the candidate index is kept in memory and the disambiguation index is on disk while in Current (2) both indexes are loaded to memory.

Language	Model	Articles	Disk	Memory
Dutch	Our build	1.1m	489MB	1.9GB
	Current (1)		2.1GB	2.4GB
	Current (2)		2.1GB	14.6GB
English	Our build	4.1m	5.2GB	11.7GB
	Current (1)		18.2GB	5.9GB

Table 2: Space requirements

3.2 Phrase Spotting

We evaluated the phrase spotting performance automatically on the first 10.000 paragraphs of a Dutch held-out data set from Wikipedia. Table 3 shows the results for the default spotting algorithm of the current DBpedia Spotlight build, as well as the language-independent FSA-based implementation and language-dependent OpenNLP-based implementation of our build.

Language	Model	Precision	Recall	F_1
Dutch	Lang. dep.	49.45	55.53	52.32
	Lang. indep.	48.26	55.01	51.42
	Current	8.26	77.17	14.92

Table 3: Phrase spotting evaluation

3.3 Disambiguation

We automatically evaluated the disambiguation performance of both builds for Dutch on 28.475 randomly selected paragraphs from the full held-out data set with only ambiguous annotations. Results are shown in Table 4. *MRR* (mean reciprocal rank) indicates n-best performance and *No URI* is the percentage of annotations in which the correct URI was not among the first 20 candidates. Note that the current build for Dutch is trained on the full Wikipedia data. Our build is trained on only the training section of the Wikipedia corpus, excluding the held-out sections. The English results are measured on an independent test corpus [4].

Language	Model	Accuracy	MRR	No URI
Dutch	Our build	0.841	0.622	0.067
Dutch	Current	0.581	0.432	0.367
English	Our build	0.851	0.797	0.074
English	Current	0.716	0.688	0.166

Table 4: Disambiguation evaluation

3.4 Internationalized Models

We created models for a number of languages among the largest Wikipedias. Table 5 presents performance values as well as the size of the respective input corpus for each of these languages. As above, we show accuracy, mean reciprocal rank and the percentage of annotations for which the correct URI was not found for the disambiguation sub-task. Additionally, we include the accuracy Acc_α of a strong baseline for each language. The baseline is to always pick the

Input		Performance			Phrase Spotting				Disambiguation			
Language	Articles	Disk	Memory	Time/par.	Type	Pr.	Re.	F_1	Acc_α	Acc	MRR	No URI
German	1.5m	1.5GB	4.2GB	113ms	T, C	43.08	50.10	46.32	0.766	0.771	0.630	0.089
French	1.3m	1.2GB	2.4GB	36ms	I, FSA	44.81	45.25	45.02	0.774	0.789	0.677	0.089
Italian	1m	944MB	1.7GB	42ms	I, FSA	47.20	50.36	48.72	0.774	0.784	0.680	0.106
Russian	991k	881MB	2GB	20ms	I, FSA	40.31	30.62	34.80	0.667	0.680	0.537	0.226
Spanish	980k	950MB	2.2GB	49ms	I, FSA	42.12	46.29	44.10	0.753	0.755	0.656	0.138
Hungarian	238k	302MB	0.9GB	14ms	I, FSA	41.85	34.07	37.56	0.809	0.833	0.564	0.095
Danish	176k	158MB	0.6GB	27ms	T, FSA	46.80	48.27	47.52	0.812	0.813	0.589	0.108

Table 5: Overview of internationalized entity extraction models

most common sense $\hat{e} = \arg\max_e \text{count}(e, s)$ using the same candidate mapping as the other algorithms. For the phrase spotting task, the table shows precision, recall and F1 score on heldout data, as well as the type of spotting algorithm used in the system:

Tokenization: `T` indicates that the system uses a supervised OpenNLP tokenization model and `I` indicates a semi-supervised tokenizer based on the `java.text` package.

Spotting algorithm: `FSA` uses a token-based finite state automaton to retrieve exact matches, `C` and `NER` indicate the use of OpenNLP models for phrase chunking and named entity recognition.

The performance values for each model are estimated on a heldout data set consisting of 6.000 paragraphs randomly selected from the input corpus. For evaluating disambiguation, we only consider ambiguous annotations and exclude disambiguation pages.

4. DISCUSSION AND OUTLOOK

In this paper, we presented some challenges encountered while collecting statistics, handling the necessary data and performing information extraction tasks that are key to the process of named entity recognition and disambiguation. We have evaluated different storage implementations, as well as approaches using different levels of language-specific knowledge, and discussed their impact on the adaptation of DBpedia Spotlight to other languages. The creation and usage of optimized custom data structures provided more flexibility in contrast to Lucene and LingPipe used in previous versions, which was reflected in the memory footprint and in the general performance.

The presented experiments show that our newly implemented methods provide improvements in phrase spotting and disambiguation accuracy as well as time performance and required space. This is due to more informed methods as well as a more integrated use of data structures between the phrase spotting and disambiguation components. Our linguistically motivated phrase spotting method provides slightly better accuracy than the language-independent method, however this improvement comes at the cost of speed and ease of adaptation to new languages. Furthermore, we demonstrated the ease of internationalization with our system by creating and evaluating models for Dutch, English, and 7 additional languages.

All of the data and source code used in the experiments are freely available online. Therefore, the experiments are reproducible with minimal effort, opening the door to strengthening collaborations within the research community, reaching out to industry adopters, and further improving the system. Demonstrations of the systems, documentation of the indexing and internationalization process as well as pointers to source code and data are available from the supporting material online.

We believe that entity recognition and linking with multilingual support will increasingly play a key role in knowledge acquisition, integration and retrieval on the Web. The ability to fuse information created and shared in different languages will provide supporting knowledge, contrasting facts as well as new information that can be seamlessly used across different geographic regions with little or no language-barriers. In this work, we provide some steps towards evaluating options for implementing such solutions in languages with a rich set of NLP resources, as well as in resource-poor languages, in an efficient and accurate manner.

5. ACKNOWLEDGMENTS

Parts of this work were funded by Google Summer of Code 2012 and by the FP7 grant Dicode (GA no. 257184). The authors would like to thank David de Boer and Gosse Bouma for help with the Dutch model. We also thank Faveeo and Globo.com for support and computing resources. Special thanks to MTA Sztaki (through Mihály Héder) for providing the necessary computing resources for creating the internationalized models.

6. REFERENCES

[1] X. Han and L. Sun. A generative entity-mention model for linking entities with knowledge base. In *Proc. of ACL: Human Language Technologies Vol. 1*, 2011.

[2] P. N. Mendes, J. Daiber, R. Rajapakse, F. Sasaki, and C. Bizer. Evaluating the impact of phrase recognition on concept tagging. In *Proc. of LREC*, 2012.

[3] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: shedding light on the web of documents. In *Proc. of I-SEMANTICS*, 2011.

[4] D. N. Milne and I. H. Witten. Learning to link with Wikipedia. In *Proc. of CIKM*, 2008.

[5] G. van Noord, G. Bouma, F. van Eynde, and D. de Kok et al. *Large Scale Syntactic Annotation of Written Dutch: Lassy*. 2009.