A Self-Supervised Approach for Extraction of Attribute-Value Pairs from Wikipedia Articles

Wladmir C. Brandão[1], Edleno S. Moura[2], Altigran S. Silva[2], and Nivio Ziviani[1]

[1] Dep. of Computer Science, Federal Univ. of Minas Gerais, Belo Horizonte, Brazil
http://www.dcc.ufmg.br
{wladmir,nivio}@dcc.ufmg.br
[2] Dep. of Computer Science, Federal Univ. of Amazonas, Manaus, Brazil
http://www.dcc.ufam.edu.br
{edleno,alti}@dcc.ufam.edu.br

Abstract. Wikipedia is the largest encyclopedia on the web and has been widely used as a reliable source of information. Researchers have been extracting entities, relationships and attribute-value pairs from Wikipedia and using them in information retrieval tasks. In this paper we present a self-supervised approach for autonomously extract attribute-value pairs from Wikipedia articles. We apply our method to the Wikipedia automatic infobox generation problem and outperformed a method presented in the literature by 21.92% in precision, 26.86% in recall and 24.29% in F1.

Keywords: Information extraction, attribute-value extraction, self-supervised extractor, Wikipedia.

1 Introduction

Wikipedia started in 2001 aiming to enable collaborative publication and dissemination of ideas and concepts on the web. Since then, people around the world share their knowledge through a platform that allows the organization of information into articles accessible and editable by anyone. Available in several languages, Wikipedia currently has a collection of over 16 million articles and over 24 million registered users. Each Wikipedia article published by one user is reviewed by others that evaluate issues of accuracy and completeness. This evaluation activity makes Wikipedia a valuable source of reliable information on the web.

The feasibility of using Wikipedia as a source of information has been shown to extract entities [28], relationships [20], and attribute-value pairs [1, 30, 31]. The information extracted from Wikipedia can be used in many information retrieval tasks, such as question answering [12, 13], query expansion [17, 19], multilingual information retrieval [23], and text categorization [3, 29]. The work in [11] illustrates a practical application of an attribute-value pairs extraction approach.

This paper presents WAVE (Wikipedia Attribute-Value Extractor), a self-supervised approach to autonomously extract attribute-value pairs from Wikipedia articles. Our approach is self-supervised in the sense that it uses a priori

available information to learn a baseline extractor and the training proceeds repeatedly by using the decisions of the extractor at step s to train the extractor at step $s+1$. WAVE learns how to extract an unlimited number of non-predefined attribute-value pairs from articles represented in the *enriched plain text format*. For example, consider a dataset "University": a possible *attribute* of this dataset could be a "name" with *value* equal to "Stanford University".

To verify the quality of the attribute-value pairs obtained we applied our approach to populate infoboxes with attribute-value pairs extracted from Wikipedia articles, a problem known as automatic infobox generation. An infobox is a special tabular structure that summarizes the content of Wikipedia articles displaying factual information in a set of attribute-value pairs. We considered as baseline the Kylin system [31], which also extracts attribute-value pairs from Wikipedia articles to generate infoboxes. We used precision, recall and F1 as measures to compare the results obtained by WAVE with the results obtained by the baseline. Experimental results show that WAVE outperforms the baseline by 21.92% in precision, 26.86% in recall and 24.29% in F1.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 presents WAVE components. Section 4 compares WAVE with the baseline. Finally, Section 5 concludes our work.

2 Related Work

Several approaches have addressed the problem of information extraction from Wikipedia. The DBPedia system [1] extracts attribute-value pairs from existing infoboxes associated with articles and turns it into an Resource Description Framework (RDF) knowledge base which can be later accessed by users. An RDF [25] extends the linking structure of the Web to use Unified Resource Identifiers (URIs) to name the relationship between things as well as the two ends of the link. This linking structure is usually referred to as a "triple".

The Yago system [28] extends WordNet [16] taxonomy using a mixed suite of heuristics to extract information from Wikipedia's category tags. It also presents a basic data model of entities and a logic-based representation language to allow queries from users. WordNet is a large lexical database for the English language that is widely used in computational linguistics and natural language processing researches.

The work in [20] uses simple heuristics developed by the authors themselves to extract relations from Wikipedia. The heuristics exploit lexical and syntactic patterns and combine them with several techniques such as anaphora resolution, full dependency parsing and subtree mining.

The Kylin system [31] extracts attribute-value pairs from articles and autonomously generates infoboxes. It uses maximum entropy classifiers [21] to associate sentences to attributes and conditional random fields (CRF)[3] [15, 26] extractors to take attribute-values pairs from sentences. In a later work, some

[3] Conditional random fields (CRF) is a framework for building probabilistic models to segment and label sequence data.

improvements were proposed in the Kylin system [30] by making it capable of dealing with rare infobox templates. First, the system refers to WordNet's ontology and aggregate attributes from parents to children infobox templates, e.g., knowing that *isA(Actor, Person)* infobox for *Actors* receives prior missing field *BirthPlace*. Second, the system apply TextRunner [10] to the web in order to retrieve additional sentences describing the same attribute-value pairs. Third, the system uses the Google search engine [9] to retrieve additional sentences describing an article. The combination of these techniques improves the recall by 2% to 9% while maintaining increasing precision.

WAVE differs from previous approaches in several ways. Differently from DBPedia[1], WAVE not only extract attribute-value pairs from existing infoboxes but it can also learn how to extract attribute-value pairs from new articles. Unlike the Yago system [28] and the approach presented in [20], WAVE can extract attribute-value pairs for an unlimited number of non-predefined attributes in addition to effectively extracts attribute-value pairs from articles.

The way Kylin system extracts attribute-value pairs is similar to WAVE. However, WAVE improves Kylin in many aspects. First, Kylin represents the content of articles using a plain text format while WAVE represents the content of articles using an *enriched plain text* format. Second, Kylin uses multiple sentences per article to generate CRF extractors, while WAVE uses only one sentence per article. Third, Kylin uses a sentence based segmentation model to generate CRF extractors while WAVE uses a window based segmentation model. It is important to note that all improvements proposed in the Kylin system by [30] can also be applied to WAVE.

3 Extracting Attribute-Value Pairs from Wikipedia

WAVE uses the content of Wikipedia articles to learn how to extract attribute-value pairs from it, building extraction models to perform the extraction. In this section we present the WAVE system. Figure 1 shows the main components of the system: Wikipedia processor, classifier, filter, and extractor, describing the learning step to obtain extraction models which are then used by WAVE to extract attribute-value pairs from Wikipedia.

3.1 Wikipedia Processor

The Wikipedia processor is responsible for extracting articles, infoboxes templates, and attribute-value pairs from Wikipedia corpus. The extracted elements compose a set of training data used by the other components. The Wikipedia processor has five steps:

1. Scan Wikipedia corpus and select articles associated with specific infobox templates. To be selected, an article must contain an infobox with the same name of a given infobox template name.

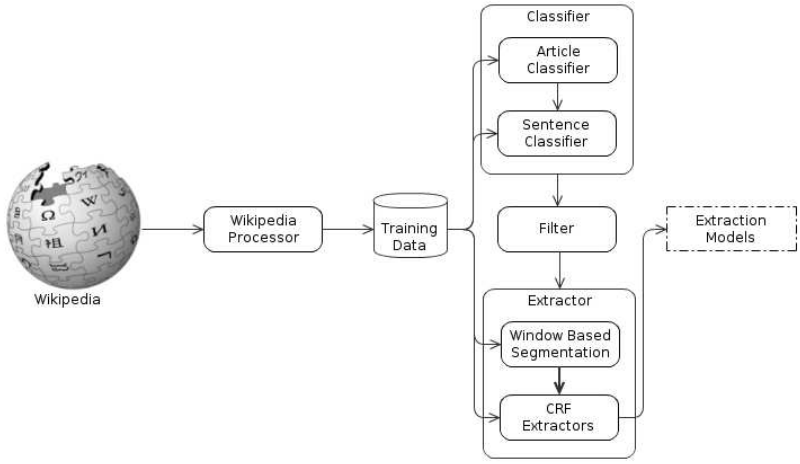


Fig. 1. The WAVE architecture.

2. Extract attribute-value pairs from the infoboxes within the selected articles and create infoboxes schemata. An infobox schema is a set of attributes for a specific infobox template.
3. Convert the content of each article from Mediawiki text format to HTML format using the Bliki engine [8], a Java API for parsing and transforming texts. Mediawiki is a Wikipedia syntax used by users to edit articles.
4. Convert the content of each article from HTML format to an *enriched plain text* format. The *enriched plain text* format considers alpha-numeric and punctuation characters. It also considers HTML tags, but discards the attributes of each tag. Figure 2 shows an example of an article content in

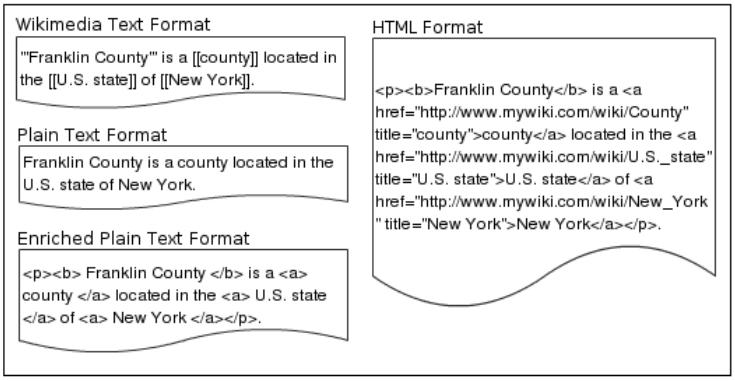


Fig. 2. Example of text formats.

Mediawiki text format, HTML format, plain-text format and *enriched plain text* format.
5. Segment the content of articles in *enriched plain text* format into sentences using OpenNLP [22], a Java API for natural language processing.

Notice that steps 3 and 4 are simply normalization procedures to make it more explicity the syntactic regularity on the text.

3.2 Classifier

Article Classifier

The Wikipedia processor component provides a set of categorized articles, i.e., articles associated with infoboxes templates. This data is used by the article classifier to learn how to associate new articles with infoboxes schemata. We need to associate each new article with exactly one infobox schema in order to determine which attributes should be extracted from the article content. We implement the article classifier using LIBSVM [5], a library for Support Vector Machines (SVM) [4, 6], which was chosen because it obtains good prediction performance for text categorization [7]. We use infobox template names, article titles, the content of articles, Wikipedia list names, and Wikipedia category tags as features for the article classifier.

Sentence Classifier

The sentence classifier is responsible for associating article sentences with article attributes. We can divide the processing of the sentence classifier into two phases:

1. Training Phase: Data provided by the Wikipedia processor component is used to build training data for the sentence classifier. For each article, sentences are associated with attributes within the infobox schema of the article. The association is based on simple term matching. If we have terms within any value of an attribute that exactly matches with terms within any sentence, they will be associated.
2. Learning Phase: A maximum entropy classifier [21] learns how to associate sentences with attributes based on training data generated in the previous phase. We use the OpenNLP Maxent library [18] for implementation of the maximum entropy classifier. This classifier was chosen in this step because it is known as a quite competitive alternative for multi-class and multi-label classification.

When a new article arrives, it is segmented into sentences. Then, the classification model learned by the sentence classifier is applied and article sentences are associated with article attributes.

3.3 Filter

The sentence classifier is a multi-class classifier and can obtain a set of sentences associated with the same attribute. The filter component is responsible for choosing the most appropriate sentence in the set.

Considering that we have the same attribute present in several infobox schema within training data, we take all the sentences related with an attribute and group them into clusters using an efficient implementation of the k-means clustering algorithm [14]. To compute the distance between sentences we represent them in a vector space and use the similarity measure as defined by the vector space model [2]. From the resultant clusters, we select one that presents the greater number of sentences from different infobox schema. This heuristic selection is based on the intuition that the most popular cluster tends to be the one that contains the best value (the best sentence) to be associated with the processed attribute. Then, we use the proximity of the cluster centroid to choose one sentence for each infobox schema. The sentence closer to the centroid is considered as the best option to fill the value of the attribute.

3.4 Extractor

The extractor is responsible for extracting attribute values from text segments and assigning them to attribute-value pairs.

Window Based Segmentation

There is a preprocessing procedure that must be done in an attempt to maximize CRF extractors performance. Each sentence associated with each attribute must be segmented into terms and a term sequence must be selected to compose the text segment to be processed by the CRF extractor.

In more details, for each sentence filtered by the filter component it is possible to determine, with the same term matching procedure used by the sentence classifier, the terms of the sentence that corresponds to the attribute value. We call this terms of "attribute-terms". Using a window size of x, we can extract from each sentence a term sequence composed of x terms before the attribute-terms (pre-terms), the attribute-terms, and x terms after the attribute-terms (post-terms). The extracted term sequences compose a training data for segmentation. It is important to note that the value x of the window size should be obtained empirically.

When a new sentence arrives, it is segmented into terms. Then, we use the similarity between the pre-terms and post-terms extracted from the sentence and the pre-terms and post-terms present in training data to select which terms will be used by CRF Extractor.

CRF Extractor

Extracting attribute values from a text can be viewed as a sequential data-labeling problem. Therefore, the choice of CRF for solving the problem is fea-

sible, since CRF is the state-of-the-art for this task. Our approach trains a different CRF extractor for each attribute and uses a well-known implementation of CRF [24]. For each attribute, the term sequences associated with it by the window based segmentation procedure is used to train the extractor. We label the pre-terms with the "pre" label, the post-terms with the "post" label and each one of the terms on the attribute-terms with three different type of labels in the following way:

1. If the attribute-terms is composed by only one term, this term is labeled with "init" label.
2. If the attribute-terms is composed by only two terms, the first term of the sequence is labeled with "init" label and the last one is labeled with "end" label.
3. If the attribute-terms is composed by more than two terms, the first term of the sequence is labeled with "init" label, the last one is labeled with "end" label and each one of the other terms is labeled with "middle" label.

Each one of the CRF extractors learn a different extraction model and use it to extract values from term sequences. The extracted value is assigned to the attribute generating an attribute-value pair.

4 Experimental Results

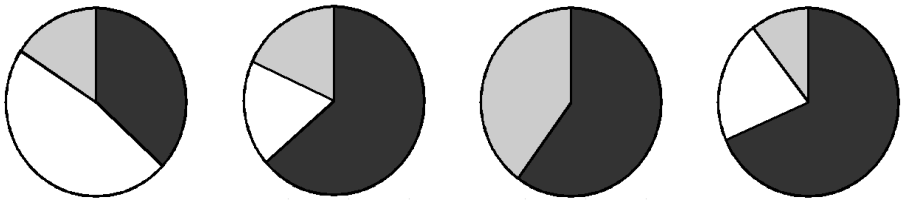
As mentioned before, we applied WAVE to address the automatic infobox generation problem and we have chosen Kylin as the baseline. In order to evaluate the performance of WAVE and the baseline, we created four datasets from the 2010.03.12 Wikipedia corpus, one for each of the following popular Wikipedia infobox templates: *U.S. County*, *Airline*, *Actor* and *University*. Table 1 shows the distribution of the 3,610 Wikipedia articles in the four datasets. The size of a dataset is the number of Wikipedia articles within the dataset.

	U.S. County	Airline	Actor	University
Dataset size	1,697	456	312	1,145

Table 1. Size of the datasets.

Some attributes do not occur frequently in Wikipedia articles and therefore were discarded. For each dataset, we discarded all attributes not present in at least 15% of its articles. The sentence classifier component also discards, for each Wikipedia article, attributes which values do not match with any word sequence in sentences within it. Table 2 shows the number of attributes extracted from Wikipedia articles, the number of attributes discarded due to low frequency and the matching procedure of the sentence classifier, and the total number of attributes actually used in experiments for each dataset.

There are three different types of attribute in each dataset: date, number, and string. Table 2 shows that we used 54 different attributes in the experiments. The most common type of attribute is the string (55.55%), followed by number (27.78%), and date attributes (16.67%). Figure 3 show the distribution of the attribute types in the datasets. We observe that only in the *U.S. County* dataset the string type is not majority. In this case, the number of numerical attributes is greater than the number of attributes of the string type.



(a) U.S. County (b) Airline (c) Actor (d) University

Fig. 3. Distribution of attribute type per dataset. String is in black, date is in gray and number is in white.

To perform the experiments, we used the 10-fold cross validation method [27]. Each dataset was randomly split in ten parts, such that, in each run, a different part was used as a training set while the remaining were used as a test set. The split on training and test sets was the same in all experiments. The final results of each experiment represent the average of the ten runs.

The performance of WAVE and the baseline was evaluated using the conventional precision, recall and F1 measures. Precision p is defined as the proportion of correctly extracted attribute-value pairs in the set of all extracted attribute-value pairs. Recall r is defined as the proportion of correctly extracted attribute-value pairs in all of the correctly attribute-value pairs in examples. $F1$ is a combination of precision and recall defined as $2pr/(p+r)$.

We performed preliminary experiments in order to empirically obtain the value of the window size x used by the window based segmentation procedure of the WAVE extractor component. We use $x = 3$ for all experiments.

Dataset	Number of Attributes			
	Extracted	Discarded		Used
		Low frequency	No matching	
U.S. County	105	75 (71.42%)	11 (10.48%)	19 (18.10%)
Airline	60	40 (66.67%)	9 (15.00%)	11 (18.33%)
Actor	45	34 (75.55%)	6 (13.34%)	5 (11.11%)
University	283	251 (88.70%)	13 (4.59%)	19 (6.71%)

Table 2. Number of extracted, discarded, and used attributes for each dataset.

As mentioned before, WAVE trains a different CRF extractor for each attribute. We perform experiments using the 10-fold cross validation method and, for each attribute, we compute the average precision, average recall, and average F1.

Table 3 shows the overall performance for WAVE and the baseline, for each attribute type. The values presented correspond to the mean average precision, mean average recall and mean average F1 for the group of attributes of each type.

Attribute Type	Precision (%)			Recall (%)			F1 (%)		
	Baseline	WAVE	Gain	Baseline	WAVE	Gain	Baseline	WAVE	Gain
Date	57.46	63.33	+10.20	52.34	57.34	+9.56	54.67	60.04	+9.82
Number	89.19	92.34	+3.53	88.16	93.04	+5.54	88.58	92.64	+4.58
String	62.58	74.37	+18.85	55.30	67.05	+21.25	58.40	70.02	+19.91

Table 3. Mean average precision, recall and F1 results achieved by WAVE and the baseline for different attribute type.

We observe that the difference in performance between WAVE and the baseline is greater in attributes of the string type. When observing the results we realized that attributes of the string type take more advantage of the textual content enrichment made by WAVE. Remember that WAVE enriches the textual content of the articles with HTML structural information, making word patterns more regular, which have increased the performance of the CRF extractors. We observe that the greater the regularity in the word patterns of the HTML tags around the attribute value to be extracted within sentences, the better the performance of the CRF extractor. The gain for numerical numbers are a bit smaller, but can be also considered as quite important, since the results of the baseline in this case were already quite high, with almost no space for improvements.

Table 4 shows the overall performance for WAVE and the baseline for the datasets. The values presented correspond to the mean average precision, mean average recall and mean average F1 for the group of attributes within each dataset.

Dataset	Precision (%)			Recall (%)			F1 (%)		
	Baseline	WAVE	Gain	Baseline	WAVE	Gain	Baseline	WAVE	Gain
U.S. County	87.73	93.85	+6.97	86.40	93.92	+8.71	87.01	93.87	+7.88
Airline	55.73	67.95	+21.92	49.75	63.11	+26.86	52.40	65.13	+24.29
Actor	65.82	75.31	+14.42	60.57	68.73	+13.48	63.09	71.60	+13.53
University	63.87	71.59	+12.08	56.12	63.33	+12.83	59.27	66.59	+12.34

Table 4. Mean average precision, recall and F1 results achieved by WAVE and the baseline for each dataset.

We observe that all WAVE results outperform the baseline values. As expected, the gain was greater in datasets with more attributes of the type string, followed by type date. This was expected, since the baseline already presents quite high quality results for numerical attributes. Again, the type string take more advantage of the textual content enrichment made by WAVE.

5 Conclusions

In this paper we have presented WAVE, a self-supervised approach to autonomously extract attribute-value pairs from Wikipedia articles. The extracted attribute-value pairs can be used in many information retrieval tasks, such as question answering, query expansion, multilingual information retrieval, and text categorization.

We applied WAVE to address the automatic infobox generation problem and we have shown that WAVE outperforms the baseline by 21.92% in precision, 26.86% in recall and 24.29% in F1. We have also shown that the greater the regularity in the word patterns of the HTML tags around the attribute value to be extracted within sentences, the better the performance of WAVE.

Future work will concentrate on improving performance of the classifier component by exploiting other Wikipedia features. We intend to analyze the impact of window size in CRF extractors and also exploit other features, incorporate improvements described by [30], expand the types of values from date, number and string to multivalued slots such as locations and person names, and measure the improvements arising from the application of our approach in some information retrieval tasks.

Acknowledgements

We thank the partial support given by the Brazilian National Institute of Science and Technology for the Web (grant MCT/CNPq 573871/2008-6), Project InfoWeb (grant MCT/CNPq/CT-INFO 550874/2007-0), Project MinGroup (grant CNPq/CT-Amazônia 575553/2008-1), CNPq Grant 30.2209/2007-7 (Edleno S. Moura), CNPq Grant 30.8528/2007-7 (Altigran S. Silva), and CNPq Grant 30.5237/02-0 (Nivio Ziviani).

References

1. S. Auer and J. Lehmann. What have innsbruck and leipzig in common? extracting semantics from wiki content. In *Proceedings of the 4th European Conference on The Semantic Web*, pages 503–517, 2007.
2. R.A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
3. S. Banerjee, K. Ramanathan, and A. Gupta. Clustering short texts using wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 787–788, 2007.

4. B.E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pages 144–152, 1992.
5. C. Chang and C. Lin. Libsvm: a library for support vector machines, 2001. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.
6. C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.
7. S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.
8. Bliki Engine. `http://code.google.com/p/gwtwiki/`.
9. Google Search Engine. `http://www.google.com/`.
10. O. Etzioni, M. Banko, S. Soderland, and D.S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, 2008.
11. R. Hahn, C. Bizer, C. Sahnwaldt, C. Herta, S. Robinson, M. Brgle, H. Dwiger, and U. Scheel. Faceted wikipedia search. In *Proceedings of the 13th International Conference of Business Information Systems*, pages 1–11, 2010.
12. R. Higashinaka, K. Dohsaka, and H. Isozaki. Learning to rank definitions to generate quizzes for interactive information presentation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 117–120, 2007.
13. M. Kaisser. The qualim question answering demo: Supplementing answers with paragraphs drawn from wikipedia. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 32–35, 2008.
14. T. Kanungo, D.M. Mount, N.S. Netanyahu, C.D. Piatko, R. Silverman, and A.Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002.
15. J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.
16. WordNet: A lexical database for English. `http://wordnet.princeton.edu/`.
17. Y. Li, W.P.R. Luk, K.S. E. Ho, and F.L.K. Chung. Improving weak ad-hoc queries using wikipedia as external corpus. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 797–798, 2007.
18. OpenNLP Maxent Library. `http://maxent.sourceforge.net/`.
19. D.N. Milne, I.H. Witten, and D.M. Nichols. A knowledge-based search engine powered by wikipedia. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 445–454, 2007.
20. D.P. Nguyen, Y. Matsuo, and M. Ishizuka. Exploiting syntatic and semantic information for relation extraction from wikipedia. In *Proceedings of the Workshop on Text-Mining & Link-Analysis*, 2007.
21. K. Nigam, J. Lafferty, and A. Mccallum. Using maximum entropy for text classification. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 61–67, 1999.
22. OpenNLP. `http://opennlp.sourceforge.net/`.
23. M. Potthast, B. Stein, and M. Anderka. Wikipedia-based multilingual retrieval model. In *Proceedings of the 30th European Conference on Informatin Retrieval Research*, pages 522–530, 2008.
24. CRF Project. `http://crf.sourceforge.net/`.

25. Resource Description Framework (RDF). `http://www.w3.org/RDF/`.
26. F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, 2003.
27. M. Stone. Cross-validation choices and assessment of statistical predictions. *Journal of the Royal Statistical Society*, B36:111–147, 1974.
28. F.M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pages 697–706, 2007.
29. P. Wang, J. Hu, H. Zeng, L. Chen, and Z. Chen. Improving text classification by using encyclopedia knowledge. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 332–341, 2007.
30. F. Wu, R. Hoffmann, and D.S. Weld. Information extraction from wikipedia: Moving down the long tail. In *Proceeding of the 14th ACM International Conference on Knowledge Discovery and Data Mining*, pages 731–739, 2008.
31. F. Wu and D.S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management*, pages 41–50, 2007.