Hypergraph Modelization of a Syntactically Annotated English Wikipedia Dump

Edmundo-Pavel Soriano-Morales, Julien Ah-Pine, Sabine Loudcher
Université de Lyon, Laboratoire ERIC, France
{edmundo.soriano-morales, julien.ah-pine, sabine.loudcher}@univ-lyon2.fr

Abstract

Wikipedia, the well known internet encyclopedia, is nowadays a widely used source of information. To leverage its rich information, already parsed versions of Wikipedia have been proposed. We present an annotated dump of the English Wikipedia. This dump draws upon previously released Wikipedia parsed dumps. Still, we head in a different direction. In this parse we focus more into the syntactical characteristics of words: aside from the classical Part-of-Speech (PoS) tags and dependency parsing relations, we provide the full constituent parse branch for each word in a succinct way. Additionally, we propose a hypergraph network representation of the extracted linguistic information. The proposed modelization aims to take advantage of the information stocked within our parsed Wikipedia dump. We hope that by releasing these resources, researchers from the concerned communities will have a ready-to-experiment Wikipedia corpus to compare and distribute their work. We render public our parsed Wikipedia dump as well as the tool (and its source code) used to perform the parse. The hypergraph network and its related metadata is also distributed.

1. Introduction and Related Work

Today, the broad reach of Wikipedia in Text Mining (TM) and Natural Language Processing (NLP) research is indisputable. Several recent approaches and tools have been conducted based on the explicit and implicit knowledge contained in it. Certainly, Wikipedia provides a common ground for researchers and developers to test and compare their results.

Wikipedia has been used as a source of valuable data as well as a common background corpus to perform experiments and compare results for diverse NLP/TM related tasks. For example, concerning the first case, in the area of Information Extraction, Wikipedia's infoboxes structured information is used in (Wu and Weld, 2010) as a valuable resource to complement and improve their open IE system. Along the same line, (Charton and Torres-Moreno, 2010) extracted metadata from Wikipedia while leveraging its internal structure in order to produce a semantically annotated corpus. Moving on to the Information Retrieval field, features extracted from Wikipedia can also help to better predict the performance of a query (Katz et al., 2014) in a given corpus. In the second case, as a background collection for experiments, a document-aligned version of English and Italian Wikipedia has been used to determine the quality between word's translations (Vulić et al., 2011).

Wikipedia, being such a popular resource already has various off-the-shelf parsed snapshots (or dumps). These parsed dumps allow researchers to focus more into their approaches than into the extraction and transformation of Wikipedia's data. We briefly describe certain relevant parses found in the literature.

A relevant Wikipedia parsed dump example comes from (Jordi Atserias and Attardi, 2008). Their work provides a balanced amount of syntactic and semantic information. In short, the dump includes each word's Part-of-Speech (PoS) tag, their dependency relations as well as the output of three different named entity recognition parsers. Additionally, they provide a graph structure that leverages Wikipedia's internal composition alongside its corresponding metadata. Nonetheless, the resource is no longer available on the original URL although it may be obtained through Yahoo's Webscope[1] datasets library. In (Flickinger et al., 2010), they perform a deep parse analysis is performed to provide detailed syntactic and semantic information. The authors leverage a previously manually annotated portion of the English Wikipedia. They extract a grammar from this portion and also train a statistical model to automatically parse the rest of Wikipedia. Even though the parse offered is deep and rich in details, the annotation labels, as well as the corpus output format, may not be convenient and easy to use because of its complexity and particular nature. (Schenkel et al., 2007) released a purely semantic XML parse that links WordNet concepts to Wikipedia pages. They focus greatly on cleaning and pre-treating Wikipedia. In this paper we do not focus as much into the cleaning of Wikipedia as already available tools can solve the task quite well for non-specific needs. Finally, there are certain Wikipedia dumps that offer the raw cleaned text without any extra subsequent parsing or analysis. Such is the case of the corpus made available by (Shaoul and Westbury, 2010). This corpus makes use of the *WikiExtractor* script (Giuseppe Attardi, 2015) to clean the Wikipedia dump.

Although the existing parses and dumps already satisfy numerous specific research needs, they have certain limitations that drove us to build our own resource: the Syntactically Annotated English Wikipedia Dump (SAEWD). Specifically, we address the following shortcomings: the lack of constituents-based tree information, the complex output formats, the limited online access and the absence of the tools used (i.e., the source code) to create the annotated corpus. In SAEWD we include the complete parse tree information for each word provided by well-known parsing tools. We store the extracted information in a simple and already existing output format. Additionally, we give open access to the parsed dump and we share our source code with the community. The code allows anyone (with programming skills) to apply our processing pipeline and build

[1] `https://webscope.sandbox.yahoo.com/`

their own particular Wikipedia parse or even to parse other text collections. Finally, we present and provide a hypergraph linguistic network for fast NLP/TM experimentation. Indeed, SAEWD aims to be used as a stepping stone for a standard Wikipedia parsed version for the largest possible set of tasks in future research.

SAEWD uses widely known English language parsing tools, namely those included in the Stanford CoreNLP suite. Aside from being accessible and regularly maintained, it provides a common set of labels (Universal Dependencies[2]) used by numerous NLP and TM experiments. Regarding SAEWD output's format, we believe that the file format we use, which follows that of (Jordi Atserias and Attardi, 2008), allows for fast reading and simple interpretation. Among other syntactical information, we provide the constituents parse branch for each word (explained in detail in Section 3.). Constituent's paths, and hence chunk's production rules, have been proved useful as a complement feature to classic text representations (Sagae and Gordon, 2009; Bergsma et al., 2012; Massung et al., 2013).

As a second contribution, we propose a hypergraph linguistic representation. Over the past few years, research on the NLP domain has been focusing on novel techniques that take advantage of the characteristics of language networks to achieve new and interesting results (Rada Mihalcea and Dragomir Radev, 2011). That is why, in addition to SAEWD, we also propose, as a proof of concept, a hypergraph representation that stores certain information found in a SAEWD in a practical way that allows for fast and effortless data extraction. This hypergraph can be indeed considered as a Linguistic Network (Choudhury and Mukherjee, 2009). It aims to facilitate the implementation of graph-based approaches by allowing researchers to jump directly into the algorithm development stage. We use a sub-sample of the Wikipedia corpus consisting of articles related to Natural Language Processing and Text Mining.

In the following sections we describe the steps we undertook to transform a Wikipedia dump into SAEWD (Section 2), we give a detailed account of the contents of SAEWD and the format in which we stored the parsed information (Section 3), then we explain the characteristics of our proposed network structure (Section 4). Lastly, we present our final comments on the nature of the work done as well as possible future work perspectives.

2. Construction of SAEWD

The three main steps we followed to build SAEWD are presented in Figure 1. Briefly, we have one input, which is the Wikipedia dump and one output which is the parsed snapshot. In the following we provide a detailed description of each of the process.

We begin the construction of the parsed corpus with the Wikipedia dump XML file obtained from the Wikipedia database[3] from early November 2014. This dump contains around 4.7 million article pages[4]. As shown in Figure 1, we apply the following processing steps in order to yield the final parsed version.

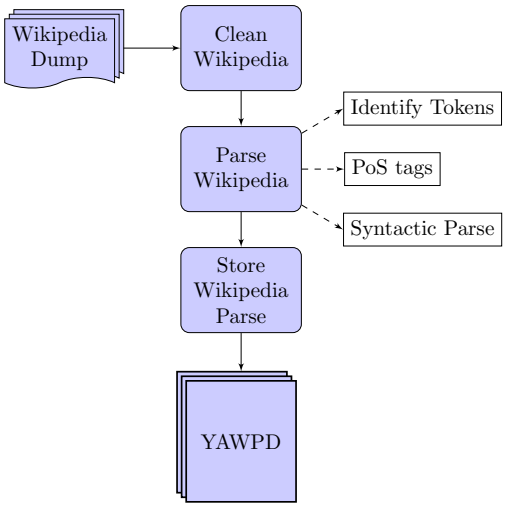


Figure 1: The tree steps we took to build SAEWD.

2.1. Cleaning Wikipedia

First, we discard Wikipedia's tables, references and lists, markup annotations and HTML tags with the *WikiExtractor* (Giuseppe Attardi, 2015) script. We used this tool to clean and split the content of the original XML file into 429 folders each one containing 100 files of approximately 300 kB. These files contain a certain number of complete Wikipedia articles which is automatically determined by WikiExtractor according to the maximum possible size assigned for each file, 300 kB in our case, thus the number of articles in each file may vary. We decided to use numerous files as well as a small size to easily read their content into memory while parsing. Having multiple small files also makes it easier to handle the multi-threading aspect of our parsing tool. We kept WikiExtractor's original folder nomenclature which assigns to each one of them a sequence of letters sorted lexicographically[5]. The files containing the cleaned text is simply named *wiki_XX* where *XX* goes from 00 to 99, as we have 100 files per folder. It is important to note that the Wikipedia articles' titles themselves are not sorted in any specific way, as it was not in the interest of our research to have them ordered. Inside each cleaned file, besides the article's text, WikiExtractor keeps the original article's URL as well as its unique Wikipedia ID within an XML-like label that also doubles as article separator.

2.2. Parsing Wikipedia

Next, once the Wikipedia dump had been cleaned, we use the Stanford CoreNLP[6] (Manning et al., 2014) analysis tools to parse all the file texts produced during the previous step. As a part of our processing pipeline, we first perform sentence segmentation, word tokenization and lemmatization. Below, we briefly describe each of the extracted attributes. We also exemplify them in detail in Section 3..

[2] `http://universaldependencies.github.io/docs/`

[3] `https://dumps.wikimedia.org/enwiki`

[4] We kept all articles available in the Wikipedia dump.

[5] We have folders named *AA*, *AB*, *AC* and so on.

[6] `http://nlp.stanford.edu/software/corenlp.shtml`

Number of tokens	1,889,769,908
Unique tokens (types)	8,761,691
Number of sentences	84,760,512
Average number of tokens per sentence	22.30

Table 1: English Wikipedia dump statistics.

- PoS tagging: We obtain the grammatical category of each word, i.e., the part-of-speech tag, using the CoreNLP default tagger, the *left3words* PoS tagging model.
- Dependency parse: this attribute consists on an extracted tree that describes the types of grammatical relations between words, i.e., the dependency-based parse tree. The analysis was performed with the Stanford's *Shift-Reduce* parser. As information representation, we use the basic dependencies scheme, as we wanted to include each one of the possible dependency relations without any collapsing between them.
- Constituents parse: the output of this analysis is a rooted tree that represents the syntactic structure of a phrase. This tree is commonly known as the constituency-based parse tree. For each word, we store its complete path in the constituency tree. Specifically, we keep all the nodes of a word's own branch from the root to the word itself. We employ the Stanford Shift-Reduce parser. This path is transformed into a single line and included in SAEWD.

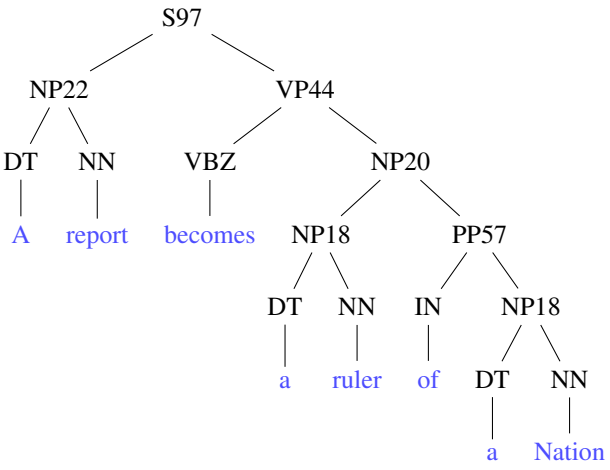
Finally, once the parsing process is complete, the parsed files are stored into individual files and thus there are as much parsed files as input Wikipedia cleaned files. The parsed files keep their original name plus the `parsed` extension, e.g., `wiki_00.parsed`. The structure within the files is described in Section 3.2.. After parsing, we found the statistics shown in Table 1.

3. SAEWD Description

In this section we describe in detail the characteristics of SAEWD.

3.1. Constituency parse storage in detail

We will use an example to better explain the storage of the constituency-based parse tree. In Figure 2 we can see the constituency parse of the phrase *A great brigand becomes a ruler of a Nation*. On the bottom of the figure, we observe the constituent's path (or branch), of the words *brigand* and *Nation*. As in any tree structure, each leaf node has a defined path from the root node to itself. In this example, the leaf containing the noun *brigand* follows the bottom-up path *NP22→S97*. *Brigand's* parent node is a Noun Phrase (NP) node which in turn comes from the root of the tree, the Sentence node *S*. We assign to each phrase chunk an identifier (22 and 97 in this case) in order to distinguish them according to their building elements as specified by the grammar rule used. In other words, a phrase chunk, e.g., a NP, a Verbal Phrase (VP), a Prepositional Phrase (PP), or other chunk defined by the grammar in CoreNLP, may be built from different types of PoS tags. Thus, again from



brigand (NN): NP22→S97
Nation (NN): NP18→PP57→NP20→VP44→S97

Figure 2: Constituency tree for the phrase *A great brigand becomes a ruler of a Nation.*

Figure 2, we see that the sentence *S97* is built both from a NP and a VP chunk. In a similar way, the noun phrase *NP18* is produced by a determinant (DT) and a noun (NN), while *NP22* is generated by a determinant, an adjective (JJ) and a noun. The identification digits are obtained from the hash code that represents each chunk object inside our Java application. For each phrase-chunk tree node, we keep the last two significative figures produced by the `hashCode`[7] Java method.

As another example, the noun *Nation* has the following bottom-up constituency path: *NP18→PP57→NP20→VP44*. Indeed, the string `NP_18,PP_57,NP_20,VP_44,S_97`, originating from the previously described path, is the information we keep about the constituency parse for each token in the Wikipedia dump.

3.2. Annotation scheme

To store the parsed text we use a scheme inspired by that used in (Jordi Atserias and Attardi, 2008). The format can be considered as a regular tsv file (i.e., the entries are separated by tab spaces) with additional metadata tags. An extract from a parsed file can be observed in Table 2.

The file includes two headers: the first one simply indicates the name of the current parse file; the second one contains the names that describe each column. The tags and columns our parsed dump contains are the following:

- Metadata tags:
  1. FILENAME: indicates the original file used to extract the current parse,
  2. %%#PAGE: denotes a new Wikipedia article, as well as its title,

[7] Java `hashCode` function description: `https://en.wikipedia.org/wiki/Java_hashCode%28%29`

FILENAME wiki_00.parsed

token	lemma	POS	constituency	head	dependency
%%#PAGE Anarchism					
⋮	⋮	⋮	⋮	⋮	⋮
%%#SEN 25 9					
A	a	DT	NP_22,S_97	3	det
great	great	JJ	NP_22,S_97	3	amod
brigand	brigand	NN	NP_22,S_97	4	nsubj
becomes	become	VBZ	VP_44,S_97	0	root
a	a	DT	NP_18,NP_20,VP_44,S_97	6	det
ruler	ruler	NN	NP_18,NP_20,VP_44,S_97	4	xcomp
of	of	IN	PP_57,NP_20,VP_44,S_97	9	case
a	a	DT	NP_18,PP_57,NP_20,VP_44,S_97	9	det
Nation	nation	NN	NP_18,PP_57,NP_20,VP_44,S_97	6	nmod

Table 2: Extract of a Wikipedia parsed file. The phrase shown is the parse result of the previous example sentence in Figure 2

PoS Tag	Token	NP				DEP				SEN
		NP_22_1	NP_20_1	NP_18_1	NP_18_2	nsubj_become	xcomp_become	nmod_ruler	amod_brigand	S_1
	brigand	1				1				1
NN	ruler		1	1			1			1
	nation		1		1			1		1
VB	becomes									1
JJ	great	1							1	1

Table 3: Brief example of the linguistic network incidence matrix of the previous used phrase. On the left side, as on the top, we can see the metadata we store for each word (rows) and each column (hyperedges). We omit the rest of the words from the example phrase for brevity.

3. %%#SEN: marks the beginning of a new sentence. It is followed by two integers: (1) the number of the current sentence, and (2), the number of tokens in the sentence.

- Parse columns for each token:

1. Token: the token itself,
2. Lemma: the token the canonical form,
3. POS: its part of speech tag,
4. Constituency: the bottom-up constituency path described before,
5. Head: the head index of the dependency relation the current token belongs to,
6. Constituency: the name of the grammatical relationship this token participates in as a dependent.

Using the example phrase introduced before (Table 2), the token *becomes* has *become* as lemma, it is a verb, thus it has *VBZ* as PoS tag, its constituency path is *VP_44,S_97*, so it belongs to the verb phrase VP44 which in turn comes from sentence S97. Finally, *becomes*, being the main verb, is in this case the grammatical root of the sentence and its head is by convention determined as zero.
Concerning the computation time, SAEWD takes around 40 hours to be produced using a general purpose laptop (Intel i7 4700MQ with 4 cores, 8 GB and Linux Mint 17 as operative system). Most of the time is taken by the parsing step.

We verified the consistency of the corpus built by analyzing a sample of 20 Wikipedia articles. The output of CoreNLP and the information contained in the corpus match.

4. Hypergraph and its Metadata

Once SAEWD is saved to disk, we leverage its information by building a linguistic network by connecting tokens according to their interaction within the Wikipedia corpus. The following section describes the network built from a sample of the information stored in SAEWD.
Given the large size of the Wikipedia corpus, we chose a sample of it to illustrate our proposed representation. We selected all the articles that are linked and that link to the Natural Language Processing Wikipedia's page.
The network is modeled as a hypergraph. Briefly, a hypergraph is a graph generalization where the main difference is that edges (named hyperedges) can link any number of vertices. Hypergraphs have been previously used to model complex networks allowing for a more complete representation of their inner dynamics (Estrada and Rodriguez-Velazquez, 2005). In our case, hyperedges allow us to group words together according to certain features, such as whether words belong to the same phrase chunk, to the same sentence, whether they share the same dependency relation or if they occur in the same context window with another and so on.
Our hypergraph provides a structure that establishes relations between tokens. These relationships can be one of two broad types: syntactical dependencies or contextual co-occurrences. The former takes into account the gram-

matical relation tokens partake in; the latter connects words according to their co-occurrences at the sentence or NP chunk level. Formally, our hypergraph is defined by a pair $G = (V, E)$, where the vertices $V = \{v_1, v_2, ..., v_n\}$ represent the set of tokens in the corpus and $E = \{e_1, e_2, ..., e_m\}$ the set of hyperedges.

Each hyperedge may be one of three types: noun phrase (*NP*), dependency (*DEP*), or sentence (*SEN*). These hyperedges represent linuistic units. The *NP* set of hyperedges contain words that appear in the same noun phrase. In the same sense, *DEP* hyperedges are the sets of words that share the same syntactic dependency relation (as dependents of a specific head). Finally, hyperedges of type *SEN* group tokens that appear in the same sentence.

Our hypergraph can be represented as a $n \times m$ incidence matrix with entries $h(i, j) = N(v_i, e_j)$ where $N(v_i, e_j)$ is the number of times $v_i \in e_j$ occurs in the corpus. That is to say, the value $h(i, j)$ correspond to the number of times a token i occurs in the same sentence, noun phrase or syntactic relation j across the entire Wikipedia corpus.

Table 3 shows the nature of the hypergraph data structure we provide using again the example phrase described in previous sections. There are two main parts to it: (1) the incidence matrix, and (2) its accompanying metadata. The matrix contains the counts of the occurrences of each word within each kind of column as stated above. The metadata is the information that describes the meaning of each row (tokens) and each column (hyperedges) in the matrix. The matrix itself contains only positive values as it is the number of times each word has occurred in each one of the hyperedges considered. The metadata, shown in Table 3, gives us, for each line and column, information about the token itself, its PoS tag, the type of column (NP,DEP, SEN), and the sub-type columns of NP and DEP, that is, what kind of noun phrase or syntactic relation we are considering. Looking at the first line of the matrix, we can know it is the token *brigand*, which is a noun (its PoS tag is *NN*). The metadata also tells us that it occurred in the first noun phrase of type *NP_22*, it took part in the nominal subject relation (*nsubj*) where the head was the lemmatized token *become*. Finally, we know it occurred in the same sentence as *ruler*, *nation*, *becomes*, *great* and the rest of the words from the phrase, which are not shown for brevity.

We store the incidence matrix (which is in fact a sparse matrix) on disk as a *Matrix Market* format file. The metadata is stored as *JSON* files. Both formats allow for an easy data lecture.

5. Conclusion and Future Work

We parsed the English Wikipedia, focusing on syntactic features namely the dependencies and constituents parse trees. We believe that the constituent membership for each token is a rich source of information for future research. The Wikipedia dump was stripped clean of tables, lists and other noisy elements, thanks to the WikiExtractor script. Our analysis was done with the CoreNLP suite which is a common tool among NLP/TM researchers and developers. It is a reliable tool that allow us to hastily and reliably process the complete corpus. The format of SAEWD follows a previous one which hopefully will allow for an easier adoption throughout the community.

We proposed and make available a hypergraph representation of a sample of the data collected in SAEWD. All our resources are available in `http://eric.univ-lyon2.fr/~psoriano/SAWD.html`.

As future research directions, we are currently working with the information contained in SAEWD, in part using the network proposed. We believe that the use of the constituency paths, coupled with the classic word co-occurrence relations, can shed light into interesting results to a number of tasks in NLP. Concerning the hypergraph, we plan to analyze the possible redundancy that may exist while adding hyperedges for each noun phrase found in the corpus. Although we get access to detailed syntactic information, the number of columns grows rapidly. Finally, a comparison between the nature of the network among different languages is also of our interest.

6. References

Bergsma, S., Post, M., and Yarowsky, D. (2012). Stylometric Analysis of Scientific Articles. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 327–337, Stroudsburg, PA, USA. Association for Computational Linguistics.

Charton, E. and Torres-Moreno, J.-M. (2010). NLGbAse: A Free Linguistic Resource for Natural Language Processing Systems. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Choudhury, M. and Mukherjee, A. (2009). The structure and dynamics of linguistic networks. In Ganguly, N., Deutsch, A., and Mukherjee, A., editors, *Dynamics On and Of Complex Networks*, Modeling and Simulation in Science, Engineering and Technology, pages 145–166. Birkhäuser Boston.

Estrada, E. and Rodriguez-Velazquez, J. A. (2005). Complex networks as hypergraphs. *arXiv preprint physics/0505137*.

Flickinger, D., Oepen, S., and Ytrestøl, G. (2010). WikiWoods: Syntacto-Semantic Annotation for English Wikipedia. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May. European Language Resources Association (ELRA).

Giuseppe Attardi. (2015). WikiExtractor. `https://github.com/attardi/wikiextractor`.

Jordi Atserias, Hugo Zaragoza, M. C. and Attardi, G. (2008). Semantically Annotated Snapshot of the English Wikipedia. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA).

Katz, G., Shtock, A., Kurland, O., Shapira, B., and Rokach, L. (2014). Wikipedia-based query performance prediction. In *Proceedings of the 37th international ACM SI-*

GIR conference on Research & development in information retrieval, pages 1235–1238. ACM.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Massung, S., Zhai, C., and Hockenmaier, J. (2013). Structural parse tree features for text representation. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, pages 9–16.

Rada Mihalcea and Dragomir Radev. (2011). *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press. Cambridge Books Online.

Sagae, K. and Gordon, A. S. (2009). Clustering Words by Syntactic Similarity Improves Dependency Parsing of Predicate-Argument Structures. In *International Conference on Parsing Technologies (IWPT-09)*, Paris, France, October.

Schenkel, R., Suchanek, F. M., and Kasneci, G. (2007). YAWN: A semantically annotated wikipedia XML corpus. In *Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 7.-9. März 2007, Aachen, Germany*, pages 277–291.

Shaoul, C. and Westbury, C. (2010). The Westbury Lab Wikipedia Corpus. `http://www.psych.ualberta.ca/~westburylab/downloads/westburylab.wikicorp.download.html`.

Vulić, I., De Smet, W., and Moens, M.-F. (2011). Identifying word translations from comparable corpora using latent topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 479–484. Association for Computational Linguistics.

Wu, F. and Weld, D. S. (2010). Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA. Association for Computational Linguistics.