DKPro WSD – A Generalized UIMA-based Framework for Word Sense Disambiguation

Tristan Miller[1] **Nicolai Erbs**[1] **Hans-Peter Zorn**[1] **Torsten Zesch**[1,2] **Iryna Gurevych**[1,2]
(1) Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
(2) Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research and Educational Information
http://www.ukp.tu-darmstadt.de/

Abstract

Implementations of word sense disambiguation (WSD) algorithms tend to be tied to a particular test corpus format and sense inventory. This makes it difficult to test their performance on new data sets, or to compare them against past algorithms implemented for different data sets. In this paper we present DKPro WSD, a freely licensed, general-purpose framework for WSD which is both modular and extensible. DKPro WSD abstracts the WSD process in such a way that test corpora, sense inventories, and algorithms can be freely swapped. Its UIMA-based architecture makes it easy to add support for new resources and algorithms. Related tasks such as word sense induction and entity linking are also supported.

1 Introduction

Word sense disambiguation, or WSD (Agirre and Edmonds, 2006)—the task of determining which of a word's senses is the one intended in a particular context—has been a core research problem in computational linguistics since the very inception of the field. Despite the task's importance and popularity as a subject of study, tools and resources supporting WSD have seen relatively little generalization and standardization. That is, most prior implementations of WSD systems have been hard-coded for particular algorithms, sense inventories, and data sets. This makes it difficult to compare systems or to adapt them to new scenarios without extensive reimplementation.

In this paper we present DKPro WSD, a general-purpose framework for word sense disambiguation which is both modular and extensible. Its *modularity* means that it makes a logical separation between the *data sets* (e.g., the corpora to be annotated, the answer keys, manually annotated training examples, etc.), the *sense inventories* (i.e., the lexical-semantic resources enumerating the senses to which words in the corpora are assigned), and the *algorithms* (i.e., code which actually performs the sense assignments and prerequisite linguistic annotations), and provides a standard interface for each of these component types. Components which provide the same functionality can be freely swapped, so that one can easily run the same algorithm on different data sets (irrespective of which sense inventory they use), or test several different algorithms on the same data set.

While DKPro WSD ships with support for a number of common WSD algorithms, sense inventories, and data set formats, its *extensibility* means that it is easy to adapt to work with new methods and resources. The system is written in Java and is based on UIMA (Lally et al., 2009), an industry-standard architecture for analysis of unstructured information. Support for new corpus formats, sense inventories, and WSD algorithms can be added by implementing new UIMA components for them, or more conveniently by writing UIMA wrappers around existing code. The framework and all existing components are released under the Apache License 2.0, a permissive free software licence.

DKPro WSD was designed primarily to support the needs of WSD researchers, who will appreciate the convenience and flexibility it affords in tuning and comparing algorithms and data sets. However, as a general-purpose toolkit it could also be used to implement a WSD module for a real-world natural language processing application. Its support for interactive visualization of the disambiguation process also makes it a powerful tool for learning or teaching the principles of WSD.

The remainder of this paper is organized as follows: In §2 we review previous work in WSD file formats and implementations. In §3 we describe

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 37–42,
Sofia, Bulgaria, August 4-9 2013.

our system and further explain its capabilities and advantages. Finally, in §4 we discuss our plans for further development of the framework.

2 Background

In the early days of WSD research, electronic dictionaries and sense-annotated corpora tended to be small and hand-crafted on an ad-hoc basis. It was not until the growing availability of large-scale lexical resources and corpora in the 1990s that the need to establish a common platform for the evaluation of WSD systems was recognized. This led to the founding of the Senseval (and later SemEval) series of competitions, the first of which was held in 1998. Each competition defined a number of tasks with prescribed evaluation metrics, sense inventories, corpus file formats, and human-annotated test sets. For each task it was therefore possible to compare algorithms against each other. However, sense inventories and file formats still vary across tasks and competitions. There are also a number of increasingly popular resources used outside Senseval and SemEval, each with their own formats and structures: examples of sense-annotated corpora include SemCor (Miller et al., 1994), MASC (Ide et al., 2010), and WebCAGe (Henrich et al., 2012), and sense inventories include VerbNet (Kipper et al., 2008), FrameNet (Ruppenhofer et al., 2010), DANTE (Kilgarriff, 2010), BabelNet (Navigli and Ponzetto, 2012), and online community-produced resources such as Wiktionary and Wikipedia. So despite attempts at standardization, the canon of WSD resources remains quite fragmented.

The few publically available implementations of individual disambiguation algorithms, such as SenseLearner (Mihalcea and Csomai, 2005), SenseRelate::TargetWord (Patwardhan et al., 2005), UKB (Agirre and Soroa, 2009), and IMS (Zhong and Ng, 2010), are all tied to a particular corpus and/or sense inventory, or define their own custom formats into which existing resources must be converted. Furthermore, where the algorithm depends on linguistic annotations such as part-of-speech tags, the users are expected to supply these themselves, or else must use the annotators built into the system (which may not always be appropriate for the corpus language or domain).

One alternative to coding WSD algorithms from scratch is to use general-purpose NLP toolkits such as NLTK (Bird, 2006) or DKPro (Gurevych et al., 2007). Such toolkits provide individual components potentially useful for WSD, such as WordNet-based measures of sense similarity and readers for the odd corpus format. However, these toolkits are not specifically geared towards development and evaluation of WSD systems; there is no unified type system or architecture which allows WSD-specific components to be combined or substituted orthogonally.

The only general-purpose dedicated WSD system we are aware of is I Can Sense It (Joshi et al., 2012), a Web-based interface for running and evaluating various WSD algorithms. It includes I/O support for several corpus formats and implementations of a number of baseline and state-of-the-art disambiguation algorithms. However, as with previous single-algorithm systems, it is not possible to select the sense inventory, and the user is responsible for pre-annotating the input text with POS tags. The usability and extensibility of the system are greatly restricted by the fact that it is a proprietary, closed-source application fully hosted by the developers.

3 DKPro WSD

Our system, DKPro WSD, is implemented as a framework of UIMA components (type systems, collection readers, annotators, CAS consumers, resources) which the user combines into a data processing pipeline. We can best illustrate this with an example: Figure 1 shows a pipeline for running two disambiguation algorithms on the Estonian all-words task from Senseval-2. UIMA components are the solid, rounded boxes in the lower half of the diagram, and the data and algorithms they encapsulate are the light grey shapes in the upper half. The first component of the pipeline is a collection reader which reads the text of the XML-formatted corpus into a CAS (a UIMA data structure for storing layers of data and stand-off annotations) and marks the words to be disambiguated (the "instances") with their IDs. The next component is an annotator which reads the answer key—a separate file which associates each instance ID with a sense ID from the Estonian EuroWordNet—and adds the gold-standard sense annotations to their respective instances in the CAS. Processing then passes to another annotator—in this case a UIMA wrapper for TreeTagger (Schmid, 1994)—which adds POS and lemma annotations to the instances.

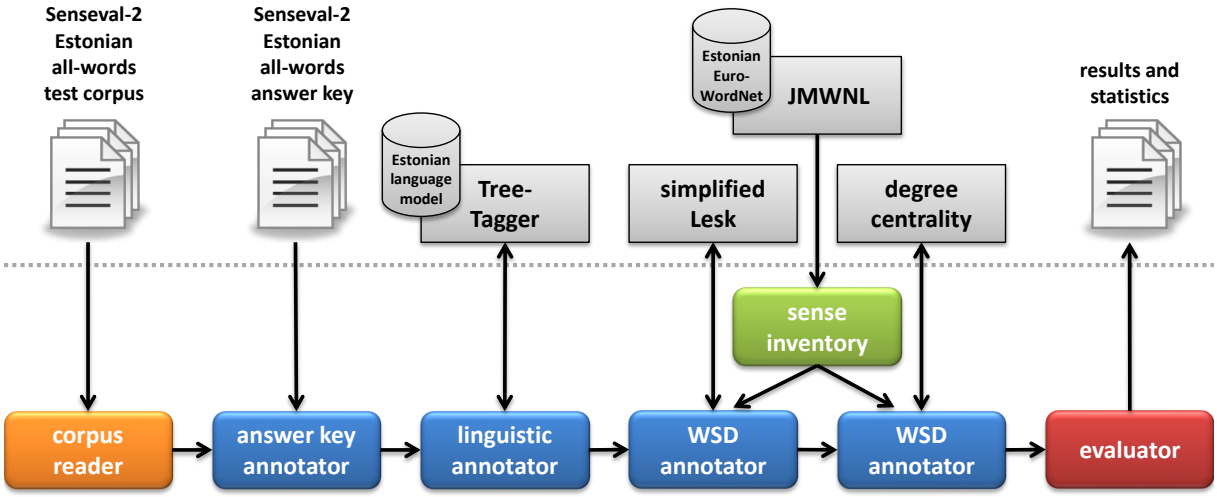

Figure 1: A sample DKPro WSD pipeline for the Estonian all-words data set from Senseval-2.

Then come the two disambiguation algorithms, also modelled as UIMA annotators wrapping non-UIMA-aware algorithms. Each WSD annotator iterates over the instances in the CAS and annotates them with sense IDs from EuroWordNet. (EuroWordNet itself is accessed via a UIMA resource which wraps JMWNL (Pazienza et al., 2008) and which is bound to the two WSD annotators.) Finally, control passes to a CAS consumer which compares the WSD algorithms' sense annotations against the gold-standard annotations produced by the answer key annotator, and outputs these sense annotations along with various evaluation metrics (precision, recall, etc.).

A pipeline of this sort can be written with just a few lines of code: one or two to declare each component and if necessary bind it to the appropriate resources, and a final one to string the components together into a pipeline. Moreover, once such a pipeline is written it is simple to substitute functionally equivalent components. For example, with only a few small changes the same pipeline could be used for Senseval-3's English lexical sample task, which uses a corpus and sense inventory in a different format and language. Specifically, we would substitute the collection reader with one capable of reading the Senseval lexical sample format, we would pass an English instead of Estonian language model to TreeTagger, and we would substitute the sense inventory resource exposing the Estonian EuroWordNet with one for WordNet 1.7.1. Crucially, none of the WSD algorithms need to be changed.

The most important features of our system are as follows:

Corpora and data sets. DKPro WSD currently has collection readers for all Senseval and SemEval all-words and lexical sample tasks, the AIDA CoNLL-YAGO data set (Hoffart et al., 2011), the TAC KBP entity linking tasks (McNamee and Dang, 2009), and the aforementioned MASC, SemCor, and WebCAGe corpora. Our prepackaged corpus analysis modules can compute statistics on monosemous terms, average polysemy, terms absent from the sense inventory, etc.

Sense inventories. Sense inventories are abstracted into a system of types and interfaces according to the sort of lexical-semantic information they provide. There is currently support for WordNet (Fellbaum, 1998), WordNet++ (Ponzetto and Navigli, 2010), EuroWordNet (Vossen, 1998), the Turk Bootstrap Word Sense Inventory (Biemann, 2013), and UBY (Gurevych et al., 2012), which provides access to WordNet, Wikipedia, Wiktionary, GermaNet, VerbNet, FrameNet, OmegaWiki, and various alignments between them. The system can automatically convert between various versions of WordNet using the UPC mappings (Daudé et al., 2003).

Algorithms. As with sense inventories, WSD algorithms have a type and interface hierarchy according to what knowledge sources they require. Algorithms and baselines already implemented include the analytically calculated random sense baseline; the most frequent sense baseline; the original, simplified, extended, and lexically expanded Lesk variants (Miller et al., 2012); various

graph connectivity approaches from Navigli and Lapata (2010); Personalized PageRank (Agirre and Soroa, 2009); the supervised TWSI system (Biemann, 2013); and IMS (Zhong and Ng, 2010). Our open API permits users to program support for further knowledge-based and supervised algorithms.

Linguistic annotators. Many WSD algorithms require linguistic annotations from segmenters, lemmatizers, POS taggers, parsers, etc. Off-the-shelf UIMA components for producing such annotations, such as those provided by DKPro Core (Gurevych et al., 2007), can be used in a DKPro WSD pipeline with little or no adaptation.

Visualization tools. We have enhanced some families of algorithms with animated, interactive visualizations of the disambiguation process. For example, Figure 2 shows part of a screenshot from the interactive running of the degree centrality algorithm (Navigli and Lapata, 2010). The system is disambiguating the three content words in the sentence "I drink milk with a straw." Red, green, and blue nodes represent senses (or more specifically, WordNet sense keys) of the words *drink*, *milk*, and *straw*, respectively; grey nodes are senses of other words discovered by traversing semantic relations (represented by arcs) in the sense inventory. The current traversal (toast%2:34:00:: to fuddle%2:34:00::) is drawn in a lighter colour. Mouseover tooltips provide more detailed information on senses. We have found such visualizations to be invaluable for understanding and debugging algorithms.

Parameter sweeping. The behaviour of many components (or entire pipelines) can be altered according to various parameters. For example, for the degree centrality algorithm one must specify the maximum search depth, the minimum vertex degree, and the context size. DKPro WSD can perform a parameter sweep, automatically running the pipeline once for every possible combination of parameters in user-specified ranges and concatenating the results into a table from which the optimal system configurations can be identified.

Reporting tools. There are several reporting tools to support evaluation and error analysis. Raw sense assignments can be output in a variety of formats (XML, HTML, CSV, Senseval answer key, etc.), some of which support colour-coding to highlight correct and incorrect assignments. The system can also compute common evaluation metrics (Agirre and Edmonds, 2006, pp. 76–80) and plot precision–recall curves for each algorithm in the pipeline, as well as produce confusion matrices for algorithm pairs. Users can specify backoff algorithms, and have the system compute results with and without the backoff. Results can also be broken down by part of speech. Figure 3 shows an example of an HTML report produced by the system—on the left is the sense assignment table, in the upper right is a table of evaluation metrics, and in the lower right is a precision–recall graph.

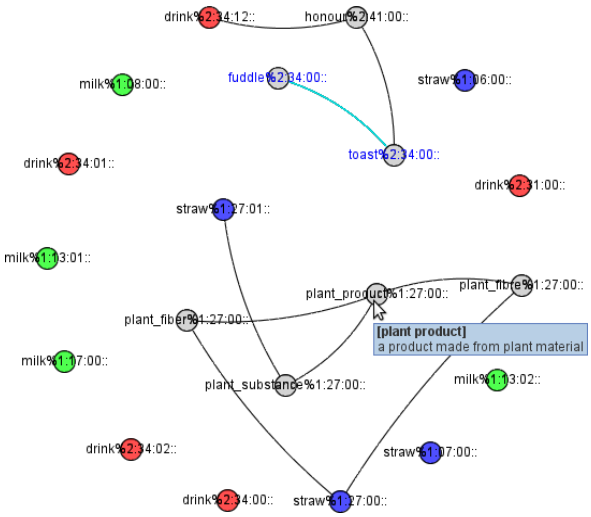

Figure 2: DKPro WSD's interactive visualization of a graph connectivity WSD algorithm.

DKPro WSD also has support for tasks closely related to word sense disambiguation:

Entity linking. Entity linking (EL) is the task of linking a named entity in a text (e.g., *Washington*) to its correct representation in some knowledge base (e.g., either *George Washington* or *Washington, D.C.* depending on the context). EL is very similar to WSD in that both tasks involve connecting ambiguous words in a text to entries in some inventory. DKPro WSD supports EL-specific sense inventories such as the list of Wikipedia articles used in the Knowledge Base Population workshop of the Text Analysis Conference (TAC KBP). This workshop, held annually since 2009, provides a means for comparing different EL systems in a controlled setting. DKPro WSD contains a reader for the TAC KBP data set, components for mapping other sense inventories to the TAC KBP inventory, and evaluation components for the

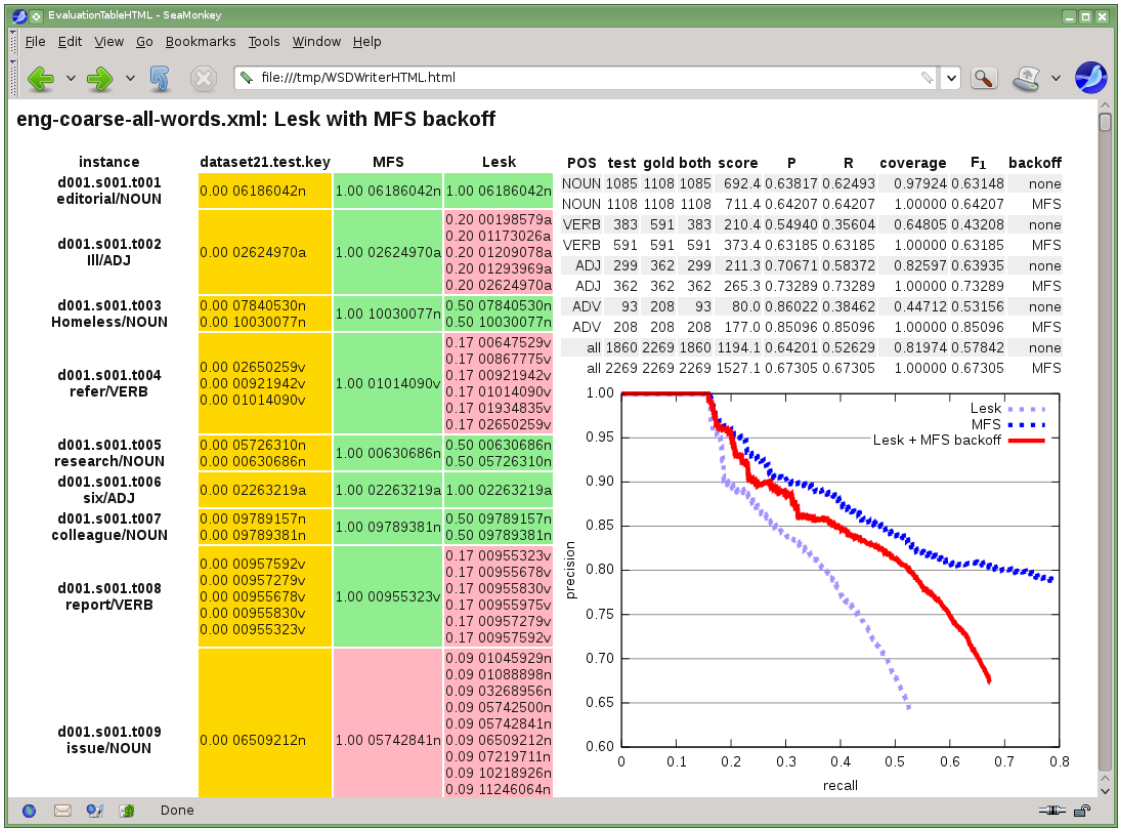

Figure 3: An HTML report produced by DKPro WSD.

official metrics. Researchers can therefore mitigate the entry barrier for their first participation at TAC KBP and experienced participants can extend their systems by making use of further WSD algorithms.

Word sense induction. WSD is usually performed with respect to manually created sense inventories such as WordNet. In word sense induction (WSI) a sense inventory for target words is automatically constructed from an unlabelled corpus. This can be useful for search result clustering, or for general applications of WSD for languages and domains for which a sense inventory is not yet available. It is usually necessary to perform WSD at some point in the evaluation of WSI. DKPro WSD supports WSI by providing state-of-the art WSD algorithms capable of using arbitrary sense inventories, including induced ones. It also includes readers and writers for the SemEval-2007 and -2013 WSI data sets.

4 Conclusions and future work

In this paper we introduced DKPro WSD, a Java- and UIMA-based framework for word sense disambiguation. Its primary advantages over existing tools are its modularity, its extensibility, and its free licensing. By segregating and providing layers of abstraction for code, data sets, and sense inventories, DKPro WSD greatly simplifies the comparison of WSD algorithms in heterogeneous scenarios. Support for a wide variety of commonly used algorithms, data sets, and sense inventories has already been implemented.

The framework is under active development, with work on several new features planned or in progress. These include implementations or wrappers for further algorithms and for the DANTE and BabelNet sense inventories. A Web interface is in the works and should be operational by the time of publication. Source code, binaries, documentation, tutorials, FAQs, an issue tracker, and community mailing lists are available on the project's website at `https://code.google.com/p/dkpro-wsd/`.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg Professorship Program under grant № I/82806.

References

Eneko Agirre and Philip Edmonds, editors. 2006. *Word Sense Disambiguation: Algorithms and Applications*. Springer.

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank for word sense disambiguation. In *Proc. EACL*, pages 33–41.

Chris Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Lang. Resour. and Eval.*, 47(1):97–122.

Steven Bird. 2006. NLTK: The natural language toolkit. In *Proc. ACL-COLING (Interactive Presentation Sessions)*, pages 69–72.

Jordi Daudé, Lluís Padró, and German Rigau. 2003. Validation and tuning of WordNet mapping techniques. In *Proc. RANLP*, pages 117–123.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

Iryna Gurevych, Max Mühlhäuser, Christof Müller, Jürgen Steimle, Markus Weimer, and Torsten Zesch. 2007. Darmstadt Knowledge Processing Repository Based on UIMA. In *Proc. UIMA Workshop at GLDV*.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY – A large-scale unified lexical-semantic resource. In *Proc. EACL*, pages 580–590.

Verena Henrich, Erhard Hinrichs, and Tatiana Vodolazova. 2012. WebCAGe – A Web-harvested corpus annotated with GermaNet senses. In *Proc. EACL*, pages 387–396.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proc. EMNLP*, pages 782–792.

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. 2010. The Manually Annotated Sub-Corpus: A community resource for and by the people. In *Proc. ACL (Short Papers)*, pages 68–73.

Salil Joshi, Mitesh M. Khapra, and Pushpak Bhattacharyya. 2012. I Can Sense It: A comprehensive online system for WSD. In *Proc. COLING (Demo Papers)*, pages 247–254.

Adam Kilgarriff. 2010. A detailed, accurate, extensive, available English lexical database. In *Proc. NAACL-HLT*, pages 21–24.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A large-scale classification of English verbs. *Lang. Resour. and Eval.*, 42(1):21–40.

Adam Lally, Karin Verspoor, and Eric Nyberg, editors. 2009. *Unstructured Information Management Architecture (UIMA) Version 1.0*. OASIS.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the TAC 2009 knowledge base population track. In *Proc. TAC*.

Rada Mihalcea and Andras Csomai. 2005. SenseLearner: Word sense disambiguation for all words in unrestricted text. In *Proc. ACL (System Demos)*, pages 53–56.

George A. Miller, Martin Chodorow, Shari Landes, Claudio Leacock, and Robert G. Thomas. 1994. Using a semantic concordance for sense identification. In *Proc. HLT*, pages 240–243.

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using distributional similarity for lexical expansion in knowledge-based word sense disambiguation. In *Proc. COLING*, pages 1781–1796.

Roberto Navigli and Mirella Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *IEEE Trans. on Pattern Anal. and Machine Intel.*, 32(4):678–692.

Roberto Navigli and Simone Paolo Ponzetto. 2012. An overview of BabelNet and its API for multilingual language processing. In Iryna Gurevych and Jungi Kim, editors, *The People's Web Meets NLP: Collaboratively Constructed Language Resources*. Springer.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2005. SenseRelate::TargetWord – A generalized framework for word sense disambiguation. In *Proc. ACL (System Demos)*, pages 73–76.

Maria Teresa Pazienza, Armando Stellato, and Alexandra Tudorache. 2008. JMWNL: An extensible multilingual library for accessing wordnets in different languages. In *Proc. LREC*, pages 28–30.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proc. ACL*, pages 1522–1531.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2010. *FrameNet II: Extended Theory and Practice*. International Computer Science Institute.

Helmud Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proc. NeMLaP*.

Piek Vossen, editor. 1998. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Springer.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A wide-coverage word sense disambiguation system for free text. In *Proc. ACL (System Demos)*, pages 78–83.