# A Platform for Named Entity Disambiguation

*Marcus Klang, Pierre Nugues*

(1) Lund University, LTH, Lund, Sweden

marcus.klang@cs.lth.se, pierre.nugues@cs.lth.se

ABSTRACT

Given a string in a text, named entity disambiguation (NED) consists in the automatic identification of the real-world reference to this string: person, organization, or country, if any. NED usually has two steps: extract the entity mentions and if a mention corresponds to a proper noun – a named entity –, possibly ambiguous and link it to a unique identifier. NED has become a core step of semantic interpretation as underlined by the new motto of the search team at Google: Things, not strings (Singhal, 2012). Most published work on named entity disambiguation focuses on English and benefits from the wide array of resources easily available for this language. In this paper, we introduce a new platform, NEDforia, to carry out named entity disambiguation for multiple languages. This platform requires minimal resources: a Wikipedia dump for the considered language and a part-of-speech tagger. It also uses Freebase (four relations), Wikidata, and Yago2 to link and map named entity candidates across languages. The NEDforia platform currently implements algorithms for Swedish and is easily extendible using a dependency injection mechanism.

KEYWORDS: Named entity disambiguation, Named entity recognition, Swedish.

## 1 Introduction

Named entity disambiguation (NED) has become a core step of semantic interpretation as underlined by the new motto of the search team at Google: Things, not strings (Singhal, 2012). IBM Watson (Ferrucci, 2012) is another example of semantic processing of text that extensively uses named entity disambiguation. NED consists usually of two steps: extract the entity mentions, usually noun phrases, and if a mention corresponds to a proper noun – a named entity –, link it to a unique identifier. Most current published work on named entity disambiguation focuses on English and benefits from the wide array of components and resources available for this language. While this component availability certainly improves the NED performance for English, a consequence is that many algorithms are not directly implementable in many other languages because of the lack of corresponding resources.

In this paper, we describe the NEDforia platform to carry out named entity disambiguation for multiple languages. NEDforia uses a Wikipedia dump for the considered language and collects automatically a list of named entities from the corpus. It then extracts the links and contexts of these entities to build disambiguation models. Given an input text, NEDforia recognizes and disambiguates the named entities and annotate them with links to their corresponding Wikipedia page. The NEDforia platform currently implements algorithms and models for Swedish and is easily extendible to other languages using a dependency injection mechanism.

## 2 Previous Work

Named entity disambiguation has been addressed using a variety of techniques. Bunescu and Paşca (2006) provided one of the first algorithms to solve it, based on the vector space model. Hoffart et al. (2011)'s paper is frequently cited as the state of the art in disambiguation. It uses an ensemble system to compute a linear combination of entity probabilities, context similarities, and entity coherences. Cucerzan (2007), Milne and Witten (2008), and Han and Zhao (2009) describe other algorithms for NED. In contrast to these previous works, multilingual support is at the core of NEDforia.

## 3 System Architecture and Implementation

We designed NEDforia to be multilingual with minimal language-specific parts. Code reuse is therefore essential across the languages as it makes the system more easily extendible. We carried out the implementation using the *dependency injection* (Fowler, 2004) pattern that solves the problem of hard-coded dependencies by making the code only depend on a dependency injector. The dependency injector is constructed once and reused multiple times. The injector can be configured to provide different concrete implementations which allow a high-level way of switching the implementation of an abstraction. The role of the injector is to provide instances of requested abstractions as well as concrete classes. The injector also injects the dependencies needed to construct these instances. We used Guice (Google, 2011) as base library on top of which we developed thread-safe constructions to be able to process indices and storage.

The system architecture consists of three major parts: A base library that serves as a foundation for the system; a toolchain that contains all the pipeline components and uses parsers, data storage components, document models from the base library; and a web server that serves as front end. The base library consists of the following packages:

**The annotation package** contains the graph-based annotation model and well as methods to query and find annotations. The annotation model supports comparison queries between

multiple outputs from different tools or runs via a versioning or variant mechanism depending on use cases. The most important annotation types are: Anchor, DependencyRelation, Entity, NamedEntity, Paragraph, Section, Sentence, Span, and Token.

**The core package** contains the dependency injection implementation based on Guice. Every specific language, e.g. Swedish, Danish, etc. is abstracted as a dependency injector, which itself inherits from a root, also a dependency injector. The specific languages are dynamically configured in the language package.

**The language package** contains the language-dependent processing tools that implement the abstracted components specified in the core package.

**The data package** contains the data abstraction tools: Lucene helpers, BerkleyDB helpers, as well as tools to convert CoNLL data to the document model.

**The disambiguation package** contains detectors and the disambiguator implementation for the NED component.

**The parser package** contains the implementation that uses Sweble to parse wiki markup.

**Import Pipeline.** NEDforia uses Wikipedia and Wikidata as a knowledge sources to extract the named entities. The encyclopedic text serves as a context for the disambiguation. The first step is to import the corpora from the XML dumps available from `http://dumps.wikimedia.org/` and segment them into pages.

We parse the dumps using the Woodstox (Codehaus Foundation, 2013) streaming XML parser and we apply a type classifier that assigns a type to every page e.g. a category, template, or article. The page types, for instance redirect, are indicated by specific tokens at the beginning of the pages such as `#REDIRECT`, or `#OMDIRIGERING`, the latter being specific to the Swedish version. To carry out the classification, the classifier uses language-dependent constants or regular expressions to match these tokens.

Once the pages are extracted and categorized, we parse them, generate an AST tree for every page, and extract text content. The parser uses Sweble and processes the resulting AST using code based on the Koshik framework (Exner and Nugues, 2014). We consider then most of the wiki markup tags, headers, paragraphs, lists tables, templates, links, and horizontal rules and we prune the remaining information. Starting from a compressed multi-streamed dump, the full implementation imports and parses the Swedish version of Wikipedia in less than 10 minutes.

**Indexing and Storage.** Once parsed, we use Lucene to index every page and we extract the named entities from page titles, links, and redirects. We store these named entities as sorted indices using Berkeley DB, where each entity is linked to the pages that contain it.

**Front end.** The front end is the interface to the application. It provides two entry points: one via a toolchain *command line interface* (CLI) and one web interface over the HTTP protocol. The tooolchain provides access to offline methods such as index construction and initial import. These are tasks that could be time consuming and need to be executed offline. The web interface provides search, a development terminal that allows users to query content within the system, and the actual interface to the disambiguator.

## 4 Entity Detection and Disambiguation

The entity detection module identifies all the strings representing named entities in a text. It uses a dictionary consisting of Wikipedia titles and anchor texts mapped to candidates. We cerated the dictionary in two steps: First through a named entity classification of the Wikipedia pages and then through an extraction of all the titles and anchor texts from classified pages. The named entity classification considers the page title and mappings to YAGO2 or Freebase (persons, organizations, locations, and events). We used Wikidata to map both YAGO2 and Freebase to the Swedish Wikipedia via English Wikipedia mappings in Wikidata in the case of YAGO2.

To detect the entities, the system splits up the text into tokens and tries to match a sequence of tokens to candidates using the dictionary. The sequence is determined by a named entity tagger, combined in a hybrid system discarding words that are not defined as proper nouns in a dictionary. As language-specific parts for Swedish, we used the Stagger POS tagger (Östling, 2013) to recognize the named entities and the Saldo dictionary. Table 1 shows an example of the surface form dictionary, where a large number of candidates represents a high level of ambiguity.

| Input | Entity description | # cand. | Sample output |
|---|---|---|---|
| *göteborg c* | Railway station in Gothenburg | 1 | wikidata:Q54326 |
| *liseberg* | Amusement park in Gothenburg | 4 | wikidata:Q1413270 |
| *haga* | District in Gothenburg | 30 | wikidata:Q1538271 |

Table 1: Entries in the detection dictionary with a short description of the sought entity.

Once we have identified the named entity strings and associated them with a list of candidate entities, we apply a disambiguation when we have more than one candidate. We implemented a variant of Bunescu and Paşca (2006)'s method.

Given a text, the user selects the detection and disambiguation methods and writes or pastes the text in the input box. NEDforia returns the annotated text in the form of hyperlinks or CoNLL tables. Figure 1 shows an annotation example, where the hyperlinks will lead the user to a Wikipedia article.


Figure 1: Example of annotation of the text: *Löftet om polisförstärkningar och mera pengar till polisområde Malmö ska enligt ett pressmeddelande från staden ha avgetts när justitieministern Beatrice Ask och rikspolischefen Bengt Svensson för några år sedan besökte Malmö. "Polisen i Malmö har ännu inte fått den utlovade resursökningen och tvingas till besparingar som i sin tur leder till minskad polisiärnärvaro, och sämre möjligheter att arbeta förebyggande", skriver nu Malmö stad i ett brev till Beatrice Ask.*

## Acknowledgments

Do not number the acknowledgment section. Do not include this section when submitting your paper for review.

## References

Bunescu, R. and Paşca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 9–16, Trento. Association for Computational Linguistics.

Codehaus Foundation (2013). Woodstox – high-performance XML processor (version 4.2.0). Last accessed: 2013-11-10.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 708–716, Prague. Association for Computational Linguistics.

Exner, P. and Nugues, P. (2014). KOSHIK: A large-scale distributed computing framework for nlp. In *Proceedings of ICPRAM 2014 – The 3rd International Conference on Pattern Recognition Applications and Methods*, pages 464–470, Angers.

Ferrucci, D. A. (2012). Introduction to "This is Watson". *IBM Journal of Research and Development*, 56(3.4):1:1 –1:15.

Fowler, M. (2004). Inversion of control containers and the dependency injection pattern. Last accessed: 2013-12-20.

Google (2011). Guice (version 3.0). Last accessed: 2013-11-10.

Han, X. and Zhao, J. (2009). Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 215 – 224.

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. (2011). Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh.

Milne, D. and Witten, I. H. (2008). Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518.

Östling, R. (2013). Stagger: an open-source part of speech tagger for Swedish. *Northern European Journal of Language Technology*, 3:1–18.

Singhal, A. (2012). Introducing the knowledge graph: things, not strings. Official Google Blog. `http://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html`. Retrieved 7 November 2013.