

Bachelor Thesis

Web-scalable Named-entity Recognition and Linking with a Wikipedia-backed Knowledge Base

Chair of Algorithms and Data Structures

Albert-Ludwigs-University Freiburg

2018-07-17/10-17

Author:	Niklas Baumert
Reviewer:	Prof. Dr. Hannah Bast
Supervisor:	Niklas Schnelle
Date of Submission:	2018-10-17

Declaration of Authorship

I hereby declare, that I am the sole author and composer of my thesis and that no other sources of learning aids, other than those listed, haven been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work. I also hereby declare that my thesis has not been prepared for another examination or assignment, either in its entirety or excerpts thereof.

Place, Date

Signature

Abstract

This thesis is about WiNERLi, a software to solve the named-entity recognition and the entity linking tasks. It does so by utilizing a knowledge base built from Wikipedia. Its major function is to provide input files for QLever¹, another system at the department.

In the following I will define the named-entity recognition and entity linking tasks and take a look at how other studies tackled the tasks. A central aspect of WiNERLi is its Wikipedia-backed knowledge base. The knowledge base is built by extracting page titles, links and infoboxes from Wikipedia pages. WiNERLi uses sub-sequences of variable lengths to determine entities by querying those against the knowledge base. I performed general tests in entity detection, entity categorization and entity linking with two datasets. Its performance in those general tests is not particularly high. It has F1-Scores of 0.3810–0.4768 in entity detection; 0.1712–0.3912 in entity categorization; and 0.2857 in entity linking.

¹<https://github.com/ad-freiburg/QLever>

Zusammenfassung

Diese Bachelorarbeit beschäftigt sich mit WiNERLi, ein Programm, das Named-Entity-Recognition und Entity-Linking durchführt. Dafür nutzt es eine auf Wikipedia basierende Wissensdatenbank. Die Hauptfunktion ist es, Eingabedateien für QLever², ein anderes System des Lehrstuhls, zu erstellen.

Im Folgenden werde ich die Named-Entity-Recognition und Entity-Linking Probleme definieren und einen Blick auf den derzeitigen Forschungsstand werfen. Der zentrale Aspekt von WiNERLi ist die Wikipedia basierende Wissensdatenbank. Diese basiert auf Daten, die aus den Seitentiteln, Links und Infoboxen von Wikipediaseiten extrahiert werden. WiNERLi nutzt Subsequenzen mit variabler Länge um Entitäten zu erkennen indem diese Subsequenzen in der Wissensdatenbank abgefragt werden. Ich habe auf zwei Datensätzen allgemeine Tests zur Entitäten Erkennung, Entitäten Kategorisierung und Entitäten Verlinkung durchgeführt. Die erzielte Leistung ist nicht hoch, mit F1-Scores von 0.3810–0.4768 bei Entitäten Erkennung; 0.1712–0.3912 bei Entitäten Kategorisierung; und 0.2857 bei Entitäten Verlinkung.

²<https://github.com/ad-freiburg/QLever>

Contents

1	Introduction	1
1.1	The Named-entity Recognition Task	1
1.2	The Entity Linking Task	2
2	Related Work	3
3	Wikipedia-backed Knowledge Base	4
4	Wikipedia Named-entity Recognition and Linking	10
4.1	Requirements	10
4.2	Implementation	12
4.2.1	Recognize Direct Mentions	13
4.2.2	Recognize Partial Mentions	13
4.2.3	Recognize Entities by Pronoun	14
4.2.4	Recognize Entities by Category	14
4.2.5	Entity Linking and Storing Results	15
5	Evaluation	16
6	Conclusion	21
6.1	WiNERLi	21
6.2	Personal Conclusions	22
	References	23

1 Introduction

This thesis is about the Wikipedia Named-entity Recognition and Linking system (WiNERLi) that I created. As the name suggests, it performs named-entity recognition (NER) and entity linking (EL) with a knowledge base (KB) built from the Wikipedia¹. NER and EL are part of the field of natural language processing (NLP). Its main goal was to produce files as input for QLever², another system of the department, for that WiNERLi parses the whole of Wikipedia.

This thesis is structured in six chapters. Chapter 1 is this introductory chapter in which I further will define the NER and EL tasks. Chapter 2 is about related work and how current state-of-the-art solutions solve the NER and EL tasks. In Chapter 3 I describe what a KB is and what the KB does in regard to the NER and EL task and how the KB for WiNERLi is created. Chapter 4 is about the requirements for WiNERLi and how it solves the NER and EL problems. In Chapter 5 I present the testing methodology and evaluate the performance of WiNERLi. Finally, Chapter 6 is the short conclusion.

1.1 The Named-entity Recognition Task

According to the definitions by Jurafsky and Martin [2009] and Zitouni [2014] the named-entity recognition task is the problem of finding and categorizing *entities* in a given text. An entity is simply anything that can be referred to by a proper noun. NER is used in, and has improved the performance of, applications in Question Answering, Machine Translation and Information Retrieval [Zitouni, 2014].

“Named entity recognition consists of the following two sub-problems: (1) recognition of named entity boundaries; (2) recognition of named entity categories (classes)” [Zitouni, 2014].

The problem with NER is, that words are ambiguous. The same word can refer to different entities of the same category [Jurafsky and Martin, 2009], this is common with people’s names. Like in the case of “John F. Kennedy” where this could mean the president or his son. But the same word can also refer to different entities of different categories [Jurafsky and Martin, 2009]. The short-hand form “JFK” can stand for the person John F. Kennedy (bringing us back to the case before) or the airport of the same name.

¹wikipedia.org

²<https://github.com/ad-freiburg/QLever>

1.2 The Entity Linking Task

According to the definitions by Raiman and Raiman [2018], Spitkovsky and Chang [2012] and Ji and Grishman [2011] the entity linking task is the problem of mention-disambiguation in documents and resolving ambiguity by “associating specific mentions in text to [entities] in a knowledge base” [Spitkovsky and Chang, 2012]. The mention “Washington” could mean either the entity “Washington D.C.” or “George Washington” [Raiman and Raiman, 2018].

WiNERLi, by default, links entities to Wikipedia pages but has means to convert resulting data to Freebase or Wikidata IDs (and back).

2 Related Work

Named-entity recognition and entity linking are relative old problems of natural language processing, but according to Johnson [2009], the field was revolutionized when scientists started transitioning from the typical grammar-based to statistical approaches (machine learning).

Most recent papers, going back to the early 2000s, are using machine learning to solve NER and EL tasks. The systems mostly differ by which underlying architecture is used. Most use a combination of bidirectional long short-term memory (LSTM) and convolution neural networks (CNN), like Santos and Guimarães [2015] and Chiu and Nichols [2015]. Ma and Hovy [2016] and Lample et al. [2016] added conditional random fields (CRF) to the bidirectional LSTM and CNN system. A graph-based system has been proposed by Han et al. [2011], and a combination of hidden Markov models and stochastic grammars proposed by Lafferty et al. [2001]

A similar design to WiNERLi has been proposed by Kazama and Torisawa [2007], as they utilize a similarly built Wikipedia-backed knowledge base, but further utilize CRF. Zhang and Ira [2009] expanded on this approach further.

For the entity linking aspect, a KB build from Wikipedia has been used by Ji and Grishman [2011], Lin et al. [2012], Han et al. [2011], Kazama and Torisawa [2007] and Zhang and Ira [2009].

3 Wikipedia-backed Knowledge Base

In this chapter I will explain what a KB is, what its benefits are in regard to NER and EL and how my KB is created.

As per definition from Section 1.2, a KB is required in the EL task to have something to map the entities to.

In the NER task gazetteers, sometimes called entity dictionaries, are frequently used to improve performance [Kazama and Torisawa, 2007; Zhang and Ira, 2009]. A gazetteer is—simply speaking—a dictionary that contains a useful mapping for the task. Which makes it similar to a knowledge base. For example, a gazetteer could contain mappings like “London” → “Location” or “Barack Obama” → “Person”. “[B]uilding and maintaining high-quality gazetteers is very time consuming” [Kazama and Torisawa, 2007]. Some automatic methods have been proposed, but they “require complicated induction of patterns or statistical methods to extract high-quality gazetteers” [Kazama and Torisawa, 2007]. One such automatic method has been proposed by Zhang and Ira [2009].

While I also use gazetteers as auxiliary information for the categories—holding a mapping from Wikipedia page title to its corresponding category—and for the gender of people—holding mapping from a persons name (which corresponds to its Wikipedia page title, too) to the respective gender—this differs greatly from the task the KB performs.

As described in Chapter 2, most state-of-the-art NER systems work with some type of machine learning. WiNERLi on the other hand works purely with a KB. This KB contains a normalized mapping from strings (characters, words or phrases) to entities. The entities are based on—and correspond to—Wikipedia pages. Which means that WiNERLi can only identify entities which have a Wikipedia page. And therefore one KB is used to solve NER (with the help of auxiliary gazetteers) and EL at the same time.

Wikipedia is a great source for this, as it is build by a voluntary community and therefore can contain pages for some rather obscure entities. There even exists a collection of unusual articles¹ with bizarre entries—that have questionable usefulness for the KB—like: “World’s littlest skyscraper”, “Smallest House in Great Britain” or “Weißwurstäquator”. The Wikipedia data is also available completely free of charge, so it is easy to work with it and enables others to easily recreate this KB. The linked inter-connectivity of the pages can be exploited, which is the topic of the rest of this chapter: the creation of the knowledge base.

The KB was first created in and for my Bachelor project, a synonym finder. The cre-

¹en.wikipedia.org/wiki/Wikipedia:Unusual_articles

```

1 invalid_chars = [ '.', ',', '!', '@', '$', '%', '^', '&', '*', '(', ')', '[', ']',
2                  '{', '}', ':', ';', '\\', '"', '\\t', '\\n' ]
3 def normalize(text):
4     clean_text = text.lower()
5     clean_text = clean_text.strip()
6     for char in invalid_chars:
7         clean_text = clean_text.replace(char, '')
8     return clean_text

```

Listing 3.1: The normalization function

ation process is heavily based on *Crosswikis* by Spitzkovsky and Chang [2012] but with some slight tweaks. For the purpose of relevance I will only discuss the relevant part of Crosswikis with regard to this thesis, but in practice the system is more complex. Crosswikis is a dictionary from strings to concepts, i.e. Wikipedia pages. They scrape “(i) English Wikipedia titles; (ii) anchor texts from English inter-Wikipedia links; (iii) anchor texts into the English Wikipedia from non-Wikipedia web-pages; and (iv) anchor texts from non-Wikipedia pages into non-English Wikipedia pages, for topics that have corresponding English Wikipedia articles” [Spitzkovsky and Chang, 2012]. This dictionary has a scoring system to determine how relevant a concept is to a string. The scoring functions are “essentially conditional probabilities” [Spitzkovsky and Chang, 2012], in this case “ratios of the number of hyperlinks into a Wikipedia URL having anchor text s and [...] the total number of anchors with text s , $S(\text{URL}|s)$, for going from strings to concepts” [Spitzkovsky and Chang, 2012].

In the case of WiNERLi the sources for information are (1) the page title; (2) links inside the page; (3) the infoboxes; of English Wikipedia pages. The process is split into two steps. In the first step, the raw data is scraped and stored in a temporary, *raw* database. The raw data is processed and stored in the true KB, which is used for NER and EL, in the second step.

In the following I will use examples for extracted mappings. For ease of read the mapped words are written as-is, but in practice they are normalized. The code listing 3.1 shows how strings, which can be just one word (ex. “Germany”), multiple words (ex. “Barack Obama”) or phrases (ex. “President of the United States”), are normalized. The given examples would normalize as follows: “Germany” → “germany”, “Barack Obama” → “barackobama” and “President of the United States” → “presidentoftheunitedstates”.

The first source are the page titles themselves. A title is stored as a direct mapping to the page itself. As an example, let’s take the page of Barack Obama. The mapping “Barack Obama” → “Barack_Obama” is stored (do note the underscore on the right-hand side, it is a *wikilink* that attached to the URL <http://en.wikipedia.org/wiki/wikilink> will lead to a valid Wikipedia page).

The second source are links. In the Wikimedia Markup Syntax² a link can be denoted

²mediawiki.org/wiki/Help:Formatting

in two forms. In the first form as `[wikilink]`, which displays a link titled “wikilink” to the page “Wikilink”, or in the second form as `[wikilink|Alternative Text]`, which displays a link titled “Alternative Text” to the page “Wikilink”. If an alternative text is given, the mapping “Alternative Text” → “Wikilink” is stored in the raw database. Otherwise, just “wikilink” → “Wikilink” is stored. Some links are ignored, mainly links to special pages like media pages or meta pages like categories. Those links are prefixed with “Image:”, “Category:” etc. Continuing the Barack Obama example, let’s take a look at some links from his article. The link `[[President of the United States]]` will map “President of the United States” → “President_of_the_United_States”. And the link `[[Democratic Party (United States)|Democratic Party]]` will map “Democratic Party” → “Democratic_Party_(United_States)”.

The third source are infoboxes. Infoboxes are Wikipedia-specific information containers on the right side of the page (see Figure 3.1). The infoboxes contain detailed information in an easy readable display (see Figure 3.3 as an example). Infoboxes are highly specialized. Figure 3.3 shows a *President*-infobox which inherits basic personal information fields from the *Person*-infobox (like the “Born” field containing birth name, date of birth and birth location). I use the type of the infobox to categorize the respective page. This yields highly specific categories. Currently only minimal further information is drawn from the infoboxes. One reason for this is, that for every infobox-category the relevant information has to be added manually to a list of attributes that will be extracted. And the second reason is, that most of the information is not relevant enough to identify an entity (for example the birth date of a person can rarely be used to uniquely identify a specific person).

Now any vivid Wikipedia user might ask why not use the categories that Wikipedia provides on the bottom of each article? And this is a valid approach which Richman and Schone [2008] and Sil and Florian [2014] have used. The problem with this in the case of WiNERLi is best shown with an example: Figure 3.2 shows every category of the *Barack Obama* Wikipedia page. Over 50 categories. Mapping these to any subset by hand is time-consuming and was not feasible to do for me. But if such a mapping exists, any Wikipedia page could be categorized, even if it doesn’t contain an infobox. Richman and Schone [2008] are resolving this by building a hierarchy of categories which gets followed until one category of a certain set of hand-picked, and already classified, categories is reached.

After the extraction process, step one, is done for every valid Wikipedia page (a valid page is an article page of some sort and no talk or media pages etc.) step two, the data processing, is started.

Barack Obama



From Wikipedia, the free encyclopedia
(Redirected from [Barrack Obama](#))

"Barack" and "Obama" redirect here. For other uses, see [Barack \(disambiguation\)](#) and [Obama \(disambiguation\)](#).

Barack Hussein Obama II (/bəˈrɑːkˈhuːseɪnˈoʊˈbɑːmə/ (listen)^[1] born August 4, 1961) is an American attorney and politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017. A member of the [Democratic Party](#), he was the first [African American](#) to serve as president. He was previously a [United States Senator](#) from [Illinois](#) and a member of the [Illinois State Senate](#).

Obama was born in 1961 in [Honolulu, Hawaii](#), two years after the territory was admitted to the Union as the 50th state. Raised largely in Hawaii, he also spent one year of his childhood in [Washington state](#) and four years in [Indonesia](#). After graduating from [Columbia University](#) in 1983, he worked as a [community organizer](#) in [Chicago](#). In 1988, he enrolled in [Harvard Law School](#), where he was the first black president of the *[Harvard Law Review](#)*. After graduating, he became a [civil rights](#) attorney and a professor, teaching [constitutional law](#) at the [University of Chicago Law School](#) from 1992 to 2004. He represented the 13th district for three terms in the [Illinois Senate](#) from 1997 to 2004, when he ran for the U.S. Senate. He received national attention in 2004 with his [March primary win](#), his well-received [July Democratic National Convention keynote address](#), and his landslide November election to the Senate. In 2008, he was nominated for president a year after [his campaign](#) began and after a [close primary campaign](#) against [Hillary Clinton](#). He was elected over [Republican John McCain](#) and was [inaugurated](#) on January 20, 2009. Nine months later, he was named the [2009 Nobel Peace Prize](#) laureate, accepting the award with the caveat that he felt there were others "far more deserving of this honor than I."

Barack Obama



44th President of the United States

In office
January 20, 2009 – January 20, 2017

Vice President [Joe Biden](#)

Preceded by [George W. Bush](#)


Succeeded by [Donald Trump](#)

Figure 3.1: A screen cap of Barack Obama's Wikipedia page with the infobox on the right side of the text.

Categories: [Barack Obama](#) | [1961 births](#) | [20th-century American writers](#) | [20th-century scholars](#) | [21st-century American politicians](#) | [21st-century American writers](#) | [21st-century scholars](#) | [Activists from Illinois](#) | [African-American feminists](#) | [African-American non-fiction writers](#) | [American Christians](#) | [American Protestants](#) | [African-American state legislators in Illinois](#) | [African-American United States presidential candidates](#) | [African-American United States Senators](#) | [American civil rights lawyers](#) | [American community activists](#) | [American feminist writers](#) | [American feminists](#) | [American legal scholars](#) | [American Nobel laureates](#) | [American people of English descent](#) | [American people of French descent](#) | [American people of German descent](#) | [American people of Irish descent](#) | [American people of Luo descent](#) | [American people of Scottish descent](#) | [American people of Swiss descent](#) | [American people of Welsh descent](#) | [American political writers](#) | [American politicians of Luo descent](#) | [Columbia University alumni](#) | [Democratic Party \(United States\) presidential nominees](#) | [Democratic Party Presidents of the United States](#) | [Democratic Party United States Senators](#) | [Grammy Award winners](#) | [Harvard Law School alumni](#) | [Illinois Democrats](#) | [Illinois lawyers](#) | [Illinois State Senators](#) | [Living people](#) | [Male feminists](#) | [Nobel Peace Prize laureates](#) | [Obama family](#) | [Occidental College alumni](#) | [Politicians from Chicago](#) | [Politicians from Honolulu](#) | [Presidents of the United States](#) | [Punahou School alumni](#) | [United States presidential candidates, 2008](#) | [United States presidential candidates, 2012](#) | [United States Senators from Illinois](#) | [University of Chicago Law School faculty](#) | [Writers from Chicago](#) | [American memoirists](#) | [Members of the American Philosophical Society](#)

Figure 3.2: The full extend of the categories on Barack Obama's Wikipedia page.

Barack Obama



44th President of the United States

In office
January 20, 2009 – January 20, 2017

Vice President [Joe Biden](#)

Preceded by [George W. Bush](#)

Succeeded by [Donald Trump](#)

**United States Senator
from Illinois**

In office
January 3, 2005 – November 16, 2008

Preceded by [Peter Fitzgerald](#)

Succeeded by [Roland Burris](#)

**Member of the Illinois Senate
from the 13th district**

In office
January 8, 1997 – November 4, 2004

Preceded by [Alice Palmer](#)

Succeeded by [Kwame Raoul](#)

Personal details

Born [Barack Hussein Obama II](#)
August 4, 1961 (age 57)
[Honolulu, Hawaii, U.S.](#)

Political party [Democratic](#)

Spouse(s) [Michelle Robinson](#) (*m.* 1992)


Children [Malia](#) · [Sasha](#)

Parents [Barack Obama Sr.](#)
[Ann Dunham](#)

Relatives [Obama family](#)

Education [Occidental College](#)
[Columbia University \(BA\)](#)
[Harvard Law School \(JD\)](#)

Awards [Nobel Peace Prize \(2009\)](#)
[Profile in Courage Award \(2017\)](#)

Signature 

Website [Office of Barack and Michelle Obama](#) [Obama Foundation](#) [White House Archives](#)

Figure 3.3: The full extend of Barack Obama's *President*-infobox

Let w be a normalized string according to Listing 3.1 stored in the raw database. For every w there exist a raw finite list of wikilinks $L^w = \{l_1^w, l_2^w, \dots, l_n^w\}$, where duplicates are possible—and very likely.

1. Take the total occurrence $to^w = |L^w|$, which is the number of wikilinks that w maps to.
2. There exist $m \leq n$ unique wikilinks in L^w . For every $l_i^w \in L^w$ calculate the occurrence $o_i^w = |\{l_j^w | l_j^w \in L^w \wedge l_j^w = l_i^w\}|$. This is how often l_i^w occurs in L^w .
3. For every wikilink $l_i^w \in L^w$ calculate $r_i^w = \frac{o_i^w}{to^w}$. This ratio is the *relevance* of the wikilink with regard to w .
4. Add the entry for w to the knowledge base: $KB(w) = \{(l_i^w, o_i^w, r_i^w) | i \in [1, m]\}$ ordered descending by r_i^w .
5. Repeat for each w .

The relevance is the measure that decides which wikilink is retrieved. This means that there are no further attempts to infer context; the most frequent wikilink is considered the best match.

Although the creation process of the KB is simple in theory, I did find some exceptions that made it difficult. First and foremost were the automatic redirects that Wikipedia employs. A redirect usually connects a well-known but maybe unofficial term for something or someone to the correct term. For example: the wikilink “Prince_Charles” redirects to the page “Charles,_Prince_of_Wales” (yes, with a comma). This problem was resolved by saving a mapping from each redirecting page to its target page and using this mapping to replace each redirection page with its target.

Another similar problem were disambiguation pages. A disambiguation page lists all Wikipedia pages of all categories which can be referred to by the term). The disambiguation page “Prince_Charles_(disambiguation)” contains links to 17 people who are or were named Prince Charles, two places and one ship. Those pages were bad results for the synonym finder, because a disambiguation should never be the synonym to a given term and they are bad results for NER and EL as they are not an entity. Luckily they all have a “disambugation-template” in the page Markup, just looking for that in the first step of the KB creation and storing those pages I was able to easily filter the pages in the second step.

Similar to disambiguation pages are “given name”-pages. For some names they contain etymology and historical usage of the name (like for Charles) while for others the page acts just as list of people with that name (see Jackie (given name) as an example). Those pages were not filtered, although they are very similar to disambiguation pages. The reason for that is how the WiNERLi checks for entities (more on that in Chapter 4). For that it needs to find an entry in the KB for partial names too, otherwise the name is considered not an entity and skipped over.

4 Wikipedia Named-entity Recognition and Linking

In this chapter I will define the requirements for WiNERLi and the ideas behind how they are fulfilled and implemented.

WiNERLi is designed to produce files that can be used as the input for QLever¹, another system from the department. For this it outputs two sets of files, *wordsfile* and *docsfile*.

The wordsfile contains four columns: `word`, `isEntity`, `recordID` and `score`. And the docsfile contains just two columns: `max record id` and `text`. Table 4.1 shows an example excerpt of a wordsfile. If `isEntity` is one, the `word` is enclosed in `< >` brackets indicating an ID. In this case Wikipedia page titles. And Table 4.2 shows an excerpt of a docsfile. The actual `text` is shortened, but this field contains the whole sentence. The `recordID` field of the wordsfile corresponds to the `max record id` field of the docsfile, meaning that the `word` is part of the corresponding sentence in the `text` field. Currently WiNERLi creates these files from the complete Wikipedia data set. The `score` field corresponds to the relevance of that entity (as described in Chapter 3).

4.1 Requirements

The following 4 requirements were given and are fulfilled by WiNERLi.

1. Recognize direct, literal mentions. This is the trivial case, where for example “Barack Obama” is part of the text.
2. Recognize partial mentions of previously mentioned entities. Continuing the example above, in following sentences “Barack” should refer to “Barack Obama”.
3. Recognize previously mentioned entities by their pronouns. “He” should refer to “Barack Obama”, after he has been mentioned before.
4. Recognize mentions of the form “the *Category*”. For example “the president” should refer to “Barack Obama”, if he was mentioned before.

¹<https://github.com/ad-freiburg/QLever>

4.1 Requirements

word	isEntity	recordID	score
Anarchism	0	1	1
<Anarchism>	1	1	0.9727
is	0	1	1
a	0	1	1
political	0	1	1
philosophy	0	1	1
<Philosophy>	1	1	0.9555
that	0	1	1
advocates	0	1	1
self	0	1	1
governed	0	1	1
societies	0	1	1
<Society>	1	1	0.7880
based	0	1	1
on	0	1	1
voluntary	0	1	1
institutions	0	1	1
<Institution>	1	1	0.7004

Table 4.1: An excerpt of the wordsfile created from *Anarchism* Wikipedia page.

max record id	text
1	Anarchism is a political philosophy that advocates [...]
2	These are often described as [...]

Table 4.2: An excerpt of the docsfile created from *Anarchism* Wikipedia page showing the beginnings of the first to sentences.

4.2 Implementation

This section is about how WiNERLi implements solutions to the requirements discussed in the previous section. Although WiNERLi does more on top of this (mainly storing the results, converting results, a web interface) but that will not be discussed further.

WiNERLi is implemented in Python 3² with some functions written in Cython³. It utilizes the excellent spaCy “industrial-strength natural language processing”⁴ framework for Python. SpaCy does the heavy lifting regarding part-of-speech tagging to identify words by their tag and the syntactic dependencies to easily split documents into sentences. For obvious reasons I do not use the NER functionality of spaCy. The following two paragraphs show one example of part-of-speech tagging. The words in the square brackets are the respective part-of-speech tag that spaCy has assigned to the word.

“Barack Hussein Obama II is an American politician who served as the 44th President of the United States from January 20, 2009, to January 20, 2017.”

“Barack[NOUN] Hussein[NOUN] Obama[NOUN] II[NOUN] is[VERB] an[DET] American[ADJ] politician[NOUN] who[NOUN] served[VERB] as[ADP] the[DET] 44th[ADJ] President[NOUN] of[ADP] the[DET] United[NOUN] States[NOUN] from[ADP] January[NOUN] 20[NUM] ,[PUNCT] 2009[NUM] ,[PUNCT] to[ADP] January[NOUN] 20[NUM] ,[PUNCT] 2017[NUM] .[PUNCT]”.

WiNERLi works on a sentence-by-sentence basis where each sentence consists of *tokens*. For each sentence sub-sequences are intelligently created—trying to fail early and skip forward as far as possible—and checked for entities. The following are two design decisions that help the intelligent creation of the sub-sequences:

1. If the first token in the sub-sequence is a punctuation symbol, start a new sub-sequence with the next token.
2. If the last token in the sub-sequence is an adposition (in, to, during, of, etc.) expand the sub-sequence by the next token.

More design decisions are discussed in their respective contexts in the following sections.

Every time an entity has been found, it is stored in the temporary storage under the respective pronoun and category, too (see the example below). If an entity has been found but at least one token of it is part of a previously found entity the entity with higher specificity (i.e. the entity with more words in it) will overwrite the previously stored entity.

In the following I will list some examples. For each of them the emphasized parts indicated detected entities and following them in square brackets are the corresponding Wikipedia pages which the entities have been linked to.

²<http://python.org>

³<http://cython.org>

⁴According to their website <http://spacy.io>

Example: Assume the entity “*Barack Obama* [Barack_Obama]” has been found. Now the temporary pronoun database stores “he” → “Barack_Obama” and the temporary category database stores “president” → “Barack_Obama”.

4.2.1 Recognize Direct Mentions

Direct mentions are the trivial case. The whole entity is part of the knowledge base. Therefore, a simple query to the KB is enough.

WiNERLi checks if the latest token in the sub-sequence is a (proper) noun or a pronoun (more on that in Section 4.2.3) and if it is, the KB is queried, otherwise a new sub-sequence is started from the next token in the sentence. If the query doesn’t yield a result, no entity was found and a new sub-sequence will be started. If the current sub-sequence contains only one token, the new sub-sequence starts at the next token. This means that the last token isn’t part of the KB. If the current sub-sequence is larger than one token, the latest token will be the start of the new sub-sequence, because it is possible for this token to be an entity in the KB on its own as it is a (proper) noun, because otherwise the sub-sequence wouldn’t have been used to query the KB.

Example: “*Barack Obama* [Barack_Obama] is an American attorney and *politician* [Politician] who served as *President of the United States* [President_of_the_United_States].”

4.2.2 Recognize Partial Mentions

This feature is only enabled for people and only works if the person was previously directly mentioned by their full name. Then each part of their name is associated with that entity until a new person with similar name(s) comes along and gets associated with the name(s). This association prevents further queries to the knowledge base.

Having the entity previously mentioned directly is required as people are difficult distinguished by just their first or last name alone.

For entities other than people wrong classifications could occur. Example (2) shows the misclassification of the second mentions of *Bank* and *Scotland* not referring to the generic financial institute and the country respectively.

WiNERLi checks all tokens of the sub-sequence against the temporary storage of partial mentions before querying against the KB. If the token is in there, the stored entity will be used and the KB will not be queried.

Example (1): “*Barack Obama* [Barack_Obama] was born in 1961 in *Honolulu* [Honolulu], *Hawaii* [Hawaii]. Raised largely in *Hawaii* [Hawaii], *Obama* [Barack_Obama] also spent one year of his childhood in the *State of Washington* [State_of_Washington].”

Example (2): “The *Bank of Scotland* [Bank_of_Scotland] is a *bank* [Bank_of_Scotland] in *Scotland* [Bank_of_Scotland].”

4.2.3 Recognize Entities by Pronoun

Pronoun recognition is limited to the three personal pronouns *he*, *she* and *it*. A gazetteer containing the mapping between people’s names and their gender is used to resolve this. Whenever an entity has been found, WiNERLi checks for the gender of the entity by looking at the gazetteer. If the entity isn’t part of this gender gazetteer, the system defaults the entity’s gender to *neutral* and associates the entity with the pronoun *it*.

The last entity mentioned for each of the three pronouns is stored temporarily. When one of the three pronouns is mentioned in the text, WiNERLi refers back to the entity that has been stored for that pronoun.

This process is not flawless, as it only uses the last mentioned entity for each pronoun. This can easily break for more complex sentences or sentences with poor grammar. Example (2) shows one such sentence.

As mentioned in Section 4.2.1, WiNERLi checks if the latest token in the sub-sequence is either a (proper) noun or a pronoun. If the token is indeed a pronoun the temporary pronoun database is checked and the stored entity is retrieved. If none is stored, a new sub-sequence is started at the next token.

Example (1): “In 1988, *Barack Obama* [Barrack_Obama] enrolled in *Harvard Law School* [Harvard_Law_School], where *he* [Barrack_Obama] was the first black *president* [President] of the *Harvard Law Review* [Harvard_Law_Review].”

Example (2): “Her *doctorate* [Doctorate] was in the field of *quantum chemistry* [Quantum_chemistry]. *It* [Quantum_chemistry] was about...”

4.2.4 Recognize Entities by Category

For this feature I had the most freedom in the sense how to find the categories for an entity. Compared to other solutions that utilize Wikipedia, which primarily use the categories that can be found at the very bottom of a page and some further classification to reduce those categories down to a smaller number [Richman and Schone, 2008; Sil and Florian, 2014]. I use the infoboxes as described in Chapter 3. This means that every article with an infobox has a category. A gazetteer containing this page-category-mapping is created beforehand. At run time, when the gazetteer is loaded, a list of all the unique categories is also created. This category list is used to identify categories in the text. Once an entity has been found, it is stored as a reference in a temporary mapping “category” → “entity”.

Every time a sub-sequence is in the temporary category mapping and specifically has “the” in-front of it, the entity stored for this category is used and no further query to the KB is required. If there’s no mapping for this category WiNERLi proceeds further.

Before the latest token in the sub-sequence is checked to be a (proper) noun or a pronoun, WiNERLi checks if the first token of the sub-sequence is not the beginning of a sentence and the token *in-front* the sub-sequence is literally the word “the”. If this is the case, the sub-sequence is treated as a category and WiNERLi checks if anything

4.2 Implementation

is stored for this category in the temporary category database. If indeed something is stored, the retrieved entity is associated with the sub-sequence, if not WiNERLi continues and checks if the sub-sequence is a normal entity.

Example (1): “During *Barack Obama’s* [Barack_Obama] first two years in office, *the president* [Barack_Obama] signed many landmark *bills* [Bill_(law)] into *law* [Law].”

Example (2): “*The Matrix* [The_Matrix] is a 1999 *science fiction* [Science_fiction] *action film* [Action_film]. *The film* [The_Matrix] depicts a dystopian *future* [Future].”

4.2.5 Entity Linking and Storing Results

In the previous sections I illustrated the recognition step of WiNERLi. By default, it does link all found entities to a corresponding Wikipedia page. Therefore, doing NER and EL in one step. But WiNERLi also supports linking to Freebase and Wikidata IDs, although in a secondary step. As of now, every sentence is processed as I described in the previous sections, the resulting data is then stored on disk. Converting this stored data from Wikipedia to Freebase or Wikidata is a second processing step, where existing Wikipedia→Freebase or Wikipedia→Wikidata mappings are loaded and used to translate the stored files to the desired format.

5 Evaluation

In this chapter I will discuss the evaluation methodology, the testing data sets and present the results.

Finding a good data set to evaluate WiNERLi was a point of great discussion. Since it isn't built to solve the general NER and EL tasks, my supervisor and I discussed for a while how I could test WiNERLi.

I use two data sets for my testing. The first one is from the website *kaggle.com*, named *Annotated Corpus for Named Entity Recognition—Feature Engineered Corpus annotated with IOB and POS tags*¹ by the user Abhinav Walia. This data set is a subset the *Groningen Meaning Bank (GMB)*², developed at the University of Groningen, which contains over 45000 sentences with parts-of-speech tags, entities and entity categories where applicable. For simplicity I will refer to this data set as *GMB-Walia* from now on.

The second data set is hand-crafted by myself. I took a handful opening paragraphs of Wikipedia pages and annotated entities and the corresponding categories. It uses the four categories *Person* (PER), *Organization* (ORG), *Location* (LOC) and *Miscellaneous* (MISC). I tried to be as unbiased as possible, but the results for this data set have to be considered with a grain of salt. Especially since Wikipedia is the base of my knowledge base too.

In Table 5.1 and Table 5.2 the statistics are listed for the Wikipedia and the GMB-Walia data set respectively. The mappings used to map the spaCy categories to the relevant categories can be seen in Table 5.3 for the GMB-Walia data set and in Table 5.4 for the Wikipedia data set. The mapping for the Wikipedia categories is too large to print in this thesis, it can be found in the `category_map.ods` file in the repository. These mappings are required for comparisons as all systems use different categories in the NER process.

	Entities						
Page	Sentences	Words	PER	ORG	LOC	MISC	Total
Konrad_Zuse	12	227	13	14	2	43	72
Caesar_cipher	6	146	2	0	0	38	40
Binary_search_algorithm	10	294	0	0	0	77	77
Total	28	667	15	14	2	158	189

Table 5.1: Statistics of the hand-annotated Wikipedia data set.

¹kaggle.com/abhinavwalia95/entity-annotated-corpus, Version 4

²gmb.let.rug.nl/

Category	Occurrences
None	1146068
Geographical Entity	58388
Organization	48094
Person	44254
Geopolitical Entity	20680
Time indicator	34789
Artifact	867
Event	612
Natural Phenomenon	300
Total Words	1354052

Table 5.2: Statistics of the GMB-Walia data set.

spaCy Category	GMB-Walia Category
PERSON	Person
NORP	Geopolitical Entity
FAC	Geographical Entity
ORG	Organization
GPE	Geopolitical Entity
LOC	Geographical Entity
PRODUCT	—
EVENT	Event
WORK_OF_ART	—
LAW	—
LANGUAGE	—
DATE	Time indicator
TIME	Time indicator
PERCENT	—
MONEY	—
QUANTITY	—
ORDINAL	—
CARDINAL	—
PER	Person
MISC	—

Table 5.3: The mapping from spaCy categories to equivalent GMB-Walia categories required to compare the results.

spaCy Category	Category
PERSON	person
NORP	organization
FAC	location
ORG	organization
GPE	organization
LOC	location
PRODUCT	misc
EVENT	misc
WORK_OF_ART	misc
LAW	misc
LANGUAGE	misc
DATE	misc
TIME	misc
PERCENT	misc
MONEY	misc
QUANTITY	misc
ORDINAL	misc
CARDINAL	misc
PER	person
MISC	misc

Table 5.4: The mapping from spaCy categories to four categories used by the hand-annotated Wikipedia data set required to compare the results.

I decided to do three tests: (1) entity detection; (2) entity categorization; (3) entity linking. For each of those I calculate *precision*, *recall* and *F1-score*. Let’s quickly define those three terms.

Let tp be the number of true-positive results, fp be the number of false-positive results and fn be the number of false-negative results for any test. We then define $Precision = \frac{tp}{tp+fp}$, $Recall = \frac{tp}{tp+fn}$ and $F1 = 2 \cdot \frac{precision \cdot recall}{precision+recall}$. Depending on which test the counting of tp , fp and fn slightly changes as follows:

1. *Entity detection* is the most basic functionality required to do NER. In this test a true-positive result is one where the tested system retrieves *any* entity for the same word as the gold solution. A false-positive result is a retrieved entity for a word where the gold solution doesn’t retrieve one. And a false-negative result is where the tested system doesn’t retrieve an entity but the gold solution does.
2. *Entity categorization* is the disambiguation process of an entity by categorizing it. In this test a true-positive result is one where the tested system categorizes the retrieved entity in the same category as the gold solution. A false-positive result is categorizing an entity where the gold solution doesn’t have a category. And a false-negative results is where the gold solution has a category for an entity but the tested system doesn’t.
3. In *Entity linking* a true-positive result is a retrieved link that matches the one of the gold solution. A false-negative result is when the tested system doesn’t retrieve a link but the gold solution does. And a false-positive result is a retrieved link where the gold solution doesn’t retrieve one.

Tests (2) and (3) are not independent from test (1). If the wrong entity is detected it is likely that the category and link of this wrong entity will not match those of the gold solution and therefore degrade the measured performance in those tests.

Tests (1) and (2) are done for both data sets, while test (3) is only done with the Wikipedia data set. For test (2) I mapped spaCy’s and WiNERLi’s categories manually to the categories used by GMB-Walia and to the four categories *Person*, *Location*, *Organization* and *Misc* for the hand-annotated Wikipedia data set.

Table 5.5, Table 5.6 and Table 5.7 present the results from each of the three tests. As spaCy doesn’t perform entity linking only WiNERLi was tested for this. WiNERLi does significantly outperform spaCy in the task of entity detection. While spaCy is better at entity categorization. The best results in each row is marked in bold.

The reasons for the worse performance of WiNERLi at test (2) are twofold. Firstly entities require an infobox to even have a category in WiNERLi, which is not always the case. Secondly, as described earlier, I had to hand-craft a mapping that reduces the over 2500 infobox categories to a comparable set. I believe that this mapping can be drastically improved and further improve the performance of WiNERLi at least slightly.

I was a bit surprised by the relative poor performance of WiNERLi at test (3), as I thought using Wikipedia both as basis for the KB and as test data would be cheating.

I think this performance is at least partially part of the problems I described in Section 4.2.3 regarding overwriting pronouns with the latest entity instead of keeping the correct one.

Entity Detection

Dataset	System	Precision	Recall	F1
Wikipedia	WiNERLi	0.5746	0.4074	0.4768
	spaCy	0.5	0.1111	0.1818
GMB-Walia	WiNERLi	1.0	0.2353	0.3810
	spaCy	1.0	0.0883	0.1622

Table 5.5: Results of the *Entity Detection* benchmark.

Entity Categorization

Dataset	System	Precision	Recall	F1
Wikipedia	WiNERLi	0.5588	0.1011	0.1712
	spaCy	0.4717	0.1330	0.2075
GMB-Walia	WiNERLi	0.5258	0.3115	0.3912
	spaCy	0.5001	0.5025	0.5013

Table 5.6: Results of the *Entity Categorization* benchmark.

Entity Linking

Dataset	System	Precision	Recall	F1
Wikipedia	WiNERLi	0.4184	0.2169	0.2857
	spaCy	N/A		

Table 5.7: Results of the *Entity Linking* benchmark.

6 Conclusion

In this final chapter I will firstly draw the conclusion on WiNERLi and secondly draw some personal conclusions working on this project and what this project taught me regarding future projects.

6.1 WiNERLi

WiNERLi as a named-entity recognition and entity linking system is still in its infancy. Although it isn't specifically build for the general NER, it can compete with spaCy. The following is a list of the main problems I see in my current implementation.

Scalable Speed: For small texts WiNERLi is fast, but it scales poorly. This is a problem as it has to work on large data sets like the whole of Wikipedia. To improve this, I would invest more time into optimizing functions and methods and also implement them in Cython. One example is the function that filters the Wikimedia Markup. It is implemented using regular expressions, that have had a case of catastrophic backtracking at least once.

Entity Recognition: It is hard to improve the NER functionality since most of the groundwork comes from spaCy. But still, there are a few things I would change:

1. Infoboxes as categories. This was a novel idea, but as the sole way to define categories for entities it is not sufficient for the general NER task. Many Wikipedia pages simply don't have them (yet), for example, some lesser known people don't even have the *Person*-infobox. While this will definitely improve in the future, I would go with the flow of other NER systems and try to retrieve the category from the broad Wikipedia categories or try to infer it from the first couple sentences.
2. More pronouns. Using *he*, *she* and *it* is fine, but using more pronouns should improve the performance further. This would be comparatively easy to implement, although a bit tedious. It would require to have a map of all handled pronouns and what gender overwrites them and whenever an entity would overwrite one of the same gender, all would be overwritten. This was not implemented because of time constraints.
3. More context. SpaCy builds *dependency trees*, which link words together. I didn't have the time to look deeper into the dependency trees to determine their usefulness. But they could help with the problem of overwriting pronouns wrongfully too.

Specificity and Entity Overwrite: This needs some deeper research and testing. As described in Chapter 4, WiNERLi does overwrite an entity when the new one contains more words. I'm not sure if this is always the best decision, so I don't have ideas to improve this yet.

6.2 Personal Conclusions

Firstly, for my next project I want to make sure I have a way to evaluate performance as early in development as possible. We struggled to find a good way to evaluate it, since it is not a generic NER/EL system.

Secondly, using poorly supported tools is not worth the hassle. In my case that was *rocksDB*. It was a really fast database when it worked, but in Python there is only one up-to-date wrapper, maintained by one person. Therefore, many bugs were not fixed as quickly as I hoped and that prompted my switch to SQLite, as it is natively supported by Python.

And last but not least, I found once again that Python's speed is a big limitation for computation-heavy tasks. I was able to improve the speed with Cython, but my knowledge of it is limited and so my Cython code was probably far from optimal. For me that means, that I need to look more into Cython, refresh my C knowledge and do more work with any compiled language, such that I can comfortably use that to solve computational-heavy problems.

References

- Chiu, J. P. and Nichols, E. (2015). Named entity recognition with bidirectional lstm-cnns.
- Han, X., Sun, L., and Zhao, J. (2011). Collective entity linking in web text: A graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 765–774. ACM.
- Ji, H. and Grishman, R. (2011). Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics.
- Johnson, M. (2009). How the statistical revolution changes (computational) linguistics. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 3–11. Association for Computational Linguistics.
- Jurafsky, D. and Martin, J. H. (2009). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*.
- Kazama, J. and Torisawa, K. (2007). Exploiting wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 698–707.
- Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition.
- Lin, T., Mausam, and Etzioni, O. (2012). Entity linking at web scale. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 84–88. Association for Computational Linguistics.
- Ma, X. and Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf.
- Raiman, J. and Raiman, O. (2018). Deeptype: Multilingual entity linking by neural type system evolution.

References

- Richman, A. E. and Schone, P. (2008). Mining wiki resources for multilingual named entity recognition. pages 1–9.
- Santos, C. d. and Guimarães, V. (2015). Boosting named entity recognition with neural character embeddings.
- Sil, A. and Florian, R. (2014). The ibm systems for english entity discovery and linking and spanish entity linking at tac 2014. In *Text Analysis Conference (TAC)*, Gaithersburg, Maryland, USA.
- Spitkovsky, V. I. and Chang, A. X. (2012). A cross-lingual dictionary for english wikipedia concepts. In *LREC*, pages 3168–3175.
- Zhang, Z. and Ira, J. (2009). A novel approach to automatic gazetteer generation using wikipedia. In *Proceedings of the 2009 Workshop on The People’s Web Meets NLP: Collaboratively Constructed Semantic Resources*. Association for Computational Linguistics.
- Zitouni, I. (2014). *Natural Language Processing of Semitic Languages*. Theory and Applications of Natural Language Processing. Springer.