



Norwegian University of
Science and Technology

Temporal Entity Linking

Beate Baier Biribakken

Master of Science in Informatics

Submission date: June 2016

Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Sammendrag

De teknologiske fremskrittene og den stadig økende mengden av elektroniske data som har utviklet seg i løpet av de siste årene har stimulert en fremvekst av forskningsfelt som undersøker hvordan man kan hente nyttig informasjon fra slike data. Denne informasjonsuthenting er en utfordring på grunn av kompleksiteten og den iboende tvetydigheten til naturlige språk.

I denne oppgaven undersøker vi om det er mulig å forbedre det polysemiske aspektet i entydiggjøring av navngitte entiteter ved å betrakte temporale data. Vi utfører en preliminær entitetslenking for å teste gjennomførbarheten til denne hypotesen ved å identifisere omtaler av personentiteter i et semantisk kodet dokumentkorpus, og forsøke å knytte disse til deres tilsvarende entiteter som befinner seg i en kunnskapsbase. Vi vil deretter vurdere om det er mulig å forbedre denne entitetslenkingen ved å vurdere den temporale dataen som er tilgjengelig i kunnskapsbasen, og i innholdet og metadataen til dokumentene i korpuset.

Vår studie tyder på at det kan være mulig å forbedre entydiggjøring av navngitte entiteter ved å vurdere temporale data, men at det trolig er lite hensiktsmessig ettersom de tilgjengelige temporale dataene i kunnskapsbasen kun er noen få eksplisitte datapunkter. Dette begrenser hva slags informasjon som kan hentes ut om en entitet, og kan egentlig bare formidle om personen var i live da dokumentet ble publisert, og kanskje utnytte informasjonen innbakt i innholdsbaserte temporale uttrykk, gitt at disse uttrykkene viser til datoer som betydelige nok til å bli registrert i kunnskapsbasen. Vår studie har også avdekket relevante aspekter ved språktvetydighet i sammenheng med entitetslenking og uthenting av temporale data som kan være nyttig i videre forskning på entydiggjøring av navngitte entiteter.

Abstract

The technological advances and an ever-growing amount of online data that has evolved over the recent years has stimulated an emergence of research fields that investigate how to extract useful information from the vast amounts of such data. This information extraction is a challenge due to the complexity and inherent ambiguity of natural language.

In this thesis we study whether it is possible to improve the polysemous aspect of named entity disambiguation in entity linking by considering temporal data. We perform a preliminary entity linking to test the feasibility of this hypothesis by identifying mentions of person entities in a semantically tagged document corpus, and attempt to link these to corresponding entities that resides in a knowledge base. We then consider whether it is possible to improve this entity linking by assessing the temporal data that is available in the knowledge base, and in the content and metadata of the documents in the corpus.

Our study indicate that it may be possible to improve named entity disambiguation by considering the temporal aspect of the data, but is likely inexpedient as the temporal data available in knowledge bases only hold a few explicit data points. This put restrictions on the information that can be extracted about an entity, and can really only convey whether the person was alive when the document was published, and perhaps exploit the information embedded in content-based temporal expressions, given that these expressions refer to dates that are remarkable enough to be registered in the knowledge base. Our study has also revealed pertinent aspects of language ambiguity in the context of entity linking and temporal information extraction that may be useful in further research on named entity disambiguation.

Acknowledgements

I would like to express my sincere gratitude to my supervisor **Heri Ramampiaro**. His advice and guidance has been invaluable to me during the work with this thesis. Throughout my studies at NTNU, his academic enthusiasm has been a true inspiration to me, and has had a great influence on the course of my academic development.

I would also like to thank **Aina Elisabeth Thunestveit** for her valuable feedback on my thesis, as well as her and **Marius Krakeli**'s generous hospitality during my visits to Trondheim.

Lastly, I would like to thank my boyfriend **Erik Lothe** for his helpful insights, and for his emotional and academic support.

Table of contents

1	Introduction	1
1.1	Motivation	3
1.2	Problem Specification	4
1.3	Project Scope	5
1.4	Report Structure	6
2	Background	7
2.1	Natural Language Processing	7
2.2	Language Ambiguity	10
2.3	Ambiguous Named Entities	13
2.4	Entity Linking	14
2.5	Information Extraction	15
2.6	Temporal Information Extraction	19
2.7	Learning Methods	22
2.8	Related work	24
3	Methodology	29
3.1	System Flow	29
3.2	Datasets	30
3.3	Preprocessing of data	37
3.4	Parsing	40
3.5	The DeepDive Framework	42
3.6	Challenges	45
4	Experiments	53
4.1	Experiment Setup	53
4.2	Results	63

5	Discussion	65
5.1	Extracting All Entities	65
5.2	Accuracy of Stanford NER	66
5.3	Data sources	68
5.4	Scalability	68
5.5	Entity Linking Quality	69
5.6	Limitations Imposed by System Design	71
5.7	Evaluation of Experiment Metrics and Results	72
5.8	Research Questions Revisited	73
6	Summary	75
6.1	Conclusion	75
6.2	Future work	76
A	Experiment Results	79
B	Parser annotators	81

List of Figures

1.1	Example of ambiguous named entities. Document contains mentions of ambiguous entities (highlighted and framed). These mentions are linked to potential target entities.	2
1.2	Example of a sentence containing three ambiguous entities and the resulting number of combinations if a naïve approach was followed. Reprinted from [1]	3
2.1	Synonymy and polysemy	13
2.2	Different representations of a constituent parse tree for the sentence "I saw the man who loves you".	16
2.3	Different representations of a dependency parse for the sentence "I saw the man who loves you". Graphs reprinted from [2]	17
2.4	Examples of explicit (transparent boxes) and relative (solid boxes) temporal expressions. Arrows indicate what kind of context information is needed to normalize the temporal expression. Reprinted from [3].	20
3.1	High-level description of system	30
3.2	Illustration of the input and output of a Knowledge Base Construction system built for paleontology. Reprinted from Zhang[4]	42
3.3	DeepDive operations. (a) Prepare datasets in relational form. (b) Generate labels using distant supervision with SQL; (c) Integrate constraints with SQL and logic functions; (d) Extract features with SQL and script languages. Reprinted from [5]	44
3.4	Screenshot from sample sentence parsed with Stanford NER	45
3.5	Screenshot of term intervals for DBpedia entry George H. W. Bush. The list is sorted by date, and the term interval is given by a combination of the list elements in the <code>dbp:termStart</code> and the list item in <code>dbp:termEnd</code> at a corresponding position.	46
3.6	Excerpt that has been parsed by SERIF and Stanford CoreNLP . .	50

4.1	Proportion of data sources from the TREC 2014 Stream corpus . .	54
4.2	The Stream Corpus Onion	55
4.3	Structure of a StreamItem	56
4.4	System flow in parallel entity linking	59
5.1	Screenshot of TAGME	70
5.2	Screenshot of entity linking	70

List of Tables

2.1	Overview of named entity categories	18
3.1	Overview of KB features	32
3.2	Summary of infobox parameters for person entities in Wikipedia . .	36
4.1	Summary of data versions	55
4.2	Description of status code responses returned by HTTP request . .	60
4.3	Confusion matrix for a binary classification problem	61
5.1	Examples	66
A.1	Precision score for when TP equals successful HTTP requests . . .	79
A.2	Distribution of HTTP response status codes for the experiments . .	80
B.1	Parser annotators	81

CHAPTER 1

Introduction

The amount of online data is growing faster than ever before. In 2013, research showed that a staggering "90 % of all the data in the world had been generated over the last two years" [6], and still growing. According to recent statistics¹, the World Wide Web² contains 4,61 billion web pages. The sheer volume, variety and velocity of the generated data can be overwhelming to comprehend. Much of this data lies buried within natural language text, tables, figures and images in web documents such as news articles, discussion forums, research articles, blog posts, reviews and social media. A common trait for all of these data elements is that they often lack structure, which essentially makes them practically infeasible to process for a computer. However, by enriching this unstructured data with semantic and syntactic information, it could be easier for computers to extract relevant information, and even to target and track specific people, events or locations.

The task of semantically and syntactically enriching large amounts of data mainly involves identifying and tagging text with grammatical categories, phrasal structural dependencies and named entities. A **named entity** is any real-world object, such as a person, an organization or a rock band. The words that has been recognized as named entities are then typically linked to a knowledge base (KB) in order to extract more information about that particular named entity. Identifying the mention of an entity and linking this entity mention to its corresponding named entity in a knowledge base is often referred to as **entity linking**. In some cases it may be unclear which entity that is being referred to in the text, either because the same name is used to refer to multiple entities, or because an entity is referred to by several names. This issue is known as the **named entity disambiguation problem**.

¹Available at www.worldwidewebsize.com. Retrieved May 9th 2016.

²The subset of web pages that is indexed by search engines.

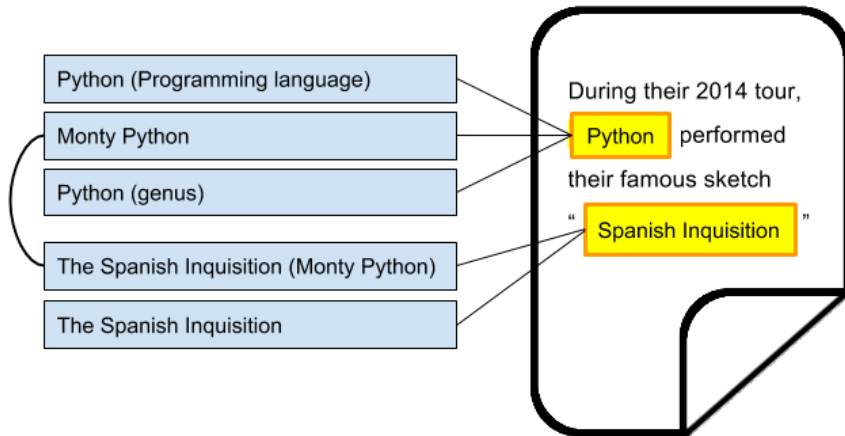


Figure 1.1: Example of ambiguous named entities. Document contains mentions of ambiguous entities (highlighted and framed). These mentions are linked to potential target entities.

Consider for instance the example in figure 1.1. Here, the mentions "Python" and "Spanish Inquisition" are ambiguous named entities because they both could refer to more than one entity. The set of entities that may be the corresponding entity for a specific entity mention are often referred to as **potential target entities**. In this case, the British comedy troupe Monty Python, the programming language Python, and the snake species Python are all potential target entities for the entity mention "Python". Although we can deduce from context that the sentence *"During their 2014 tour, Python performed their famous sketch "Spanish Inquisition" . . ."* refers to the entity "Monty Python" and the entity "The Spanish Inquisition (Monty Python)", the example demonstrates that a name is not necessarily a unique identifier for an entity. Other examples include:

- **People could be named after other people**, e.g. *George Bush Jr.* was named after his father *George Bush Sr.*
- **A disease could be named after its diagnostician**, e.g. *Down's Syndrome* was named after the British doctor *John Langdon Down*.
- **A band could be named after a historical person**, e.g. The Scottish band *Franz Ferdinand* was named after the Austrian archduke *Franz Ferdinand*.

1.1 Motivation

The explosive growth of online data has formed a basis for research fields that specialize in extracting and potentially utilizing the information embedded in the data. Systems that exploit this information is often afflicted by issues that is caused by ambiguous entities. If ambiguous entities occur, there currently are no sustainable solutions to cope with them, which consequently produce a bottleneck because the disambiguation process is likely to require human interference in order to produce quality results. The named entity disambiguation problem is a fundamental challenge in such situations, as well as a number of other domains and research fields, because it is caused by the inherent ambiguity and context-dependency of natural language.

Page played the hit Kashmir on his uniquely tuned Les Paul.

500 x 50 x 5

= 125.000 possible candidate combinations

Figure 1.2: Example of a sentence containing three ambiguous entities and the resulting number of combinations if a naïve approach was followed. Reprinted from [1]

One approach for solving the named entity disambiguation problem in entity linking could be to enumerate all possible combinations of potential target entities for each ambiguous entity [1]. This would however be a naïve approach considering that a single sentence with only a few ambiguous entity mentions alone could exceed 100.000 combinations, as demonstrated in figure 1.2. On a larger scale, it is clear that this would not be a feasible solution, and emphasizes the importance of investigating the named entity disambiguation problem in a broader context.

Solving the named entity disambiguation problem is likely to increase the performance in existing systems, and could potentially lead to scientific breakthroughs in research fields that are affected by issues caused by ambiguous entities. It could also present new opportunities and lead to new research areas and real-world applications. For example:

- A company could combine a database of their product selection with a specialized knowledge base to automate typical corporate customer care tasks such as inquiries and complaints.

- Editorial staffs and research teams could use and generate embedded work of references to look up information about specific entities during research or quality checks.
- In research areas that are affected by the named entity disambiguation problem, the resource intensive disambiguation step could be completely removed, which would dramatically improve performance in systems that currently is impeded by named entity disambiguation.

It is also important to consider that a valuable byproduct of named entity disambiguation is that for the entities to be recognized, the language must be processed and enriched with semantic and structural information. This enrichment makes it possible to transform information in one language into language-independent knowledge, which could open up a world of knowledge and furthermore be an important aid in third world countries where access to updated knowledge is scarce.

1.2 Problem Specification

Most entities are linked to temporal data in one way or another: The date for when a band was formed or a person was born, the release date for a movie or a specific piece of software, or the time interval for a music festival. Even the medium that conveys the information about the entity is usually associated with metadata such as document creation time, time of publishing and similar temporal information.

By taking into account the temporal data that is affiliated with specific entities or documents, we may be able to determine if the document contains relevant new information about specific entities or information that can be linked to specific entities based on the available temporal data.

Although several research efforts have been made into both entity linking and temporal information retrieval, few studies have focused on temporal entity linking. To the best of our knowledge, this is the first study that considers the polysemous aspect of the named entity disambiguation problem in the view of temporal data.

The main research goal of this thesis is to investigate whether it is possible to improve named entity disambiguation by considering the temporal aspect of the data.

This thesis will primarily focus on disambiguation between multiple entities that can be denoted by the same name. To achieve our goal, there are several aspects that must be studied, which can be condensed into the following three research questions:

- RQ1** What kind of temporal data about entities is possible to obtain?
- RQ2** How can temporal data be exploited to disambiguate ambiguous entities?
- RQ3** Is it expedient to use temporal data for named entity disambiguation?

To achieve our goal, we developed a system which design is largely based on Stanford University's contribution [5] in the Knowledge Base Population (KBP) track at the Text Analysis Conference (TAC) in 2014, where Stanford achieved the highest overall score with respect to all the evaluation metrics [7]. This system was reimplemented and altered to achieve our research goal. Our system design will be further detailed in chapter 3.

1.3 Project Scope

This section present the scope of the project, and briefly describe the preconditions and constraints that provides a framework for the system implementation.

1.3.1 Language

This project will only consider the subset of resources in English. This is mainly because the majority of documents in our available resources are in English, but also because the tools used to analyze the data is especially adapted to the English language. By excluding all languages but English, we can reduce potential noise that could have been introduced by the lack of one-to-one correspondence between words in different languages. The linguistic aspect of the thesis will mainly be directed towards the semantic and syntactical properties of the language. It will not focus on techniques that regards the actual alterations of text, such as spelling correction and stemming.

1.3.2 Data

Due to constraints imposed by time and economy, the project will employ existing datasets that are balanced, freely available and easily accessible. The focus will be

on raw text data, and does not include rich content such as images, tables, figure, video or audio. The content will stem from several different data sources, including blogs, discussion forums, research articles, news articles and reviews, to ensure a various, heterogeneous and representative dataset. The trustworthiness of the data sources is beyond the scope of this thesis.

1.3.3 Resource Constraints

This project was granted one virtual machine with 4 cores, 8 GB RAM and 2 TB storage, limiting tests of scalability and the amount of data that could be processed.

1.4 Report Structure

The remainder of the report is structured as follows. First, **chapter 2** introduce the domain knowledge and state-of-the-art related work, and provide the reader with a context and fundamental understanding of the field of study. **Chapter 3** present tools and frameworks that was used in the development of the system, and challenges that was encountered during development. **Chapter 4** describe the experiment design, methods and results. **Chapter 5** analyze the results from the experiments and discuss the significance of the results. Lastly, **chapter 6** conclude with a summary drawn from the discussion of the results, and outline some pertinent improvements for future work.

2.1 Natural Language Processing

The natural languages as we know them today has evolved over thousands of years, even before alphabets existed, and before literacy became a common norm in society. It should therefore not come as a surprise that natural languages are characterized by local peculiarities in terms of pronunciation, alphabets, language structure, semantics, and vocabulary. Natural languages are typically also highly context-dependent because humans often use the language in a particular situation, and thus does not need to be explicit in how they express themselves. As a result, natural languages are inherently ambiguous, complex and often difficult to fully comprehend, which is the main reason why extracting information from natural language text remains a challenge in Natural Language Processing.

2.1.1 History of Natural Language Processing

Natural Language Processing (NLP) is a research field with roots in artificial intelligence and computational linguistics pertaining to the interactions between natural languages and computers. Although the computer is a relatively modern device, the very idea of a system for translating words across languages was proposed as early as the seventeenth century by philosophers such as John Wilkins, Gottfried Leibniz and René Descartes, resulting in mechanical dictionaries that by some was regarded as genuine precursors of machine translation.

During World War II there was a particular interest in how computers could be applied in practice to solve real-world problems because of the field's inherent potential strategic advantages. A fortunate byproduct of this was an increased investment in computer science, which created an environment of growth for research

that could exploit the ever growing amount of computational power, such as cryptography and artificial intelligence. In his essay "Intelligent Machinery" (1948), the artificial intelligence pioneer Alan Turing suggested a number of ways in which computers could demonstrate their intelligence, including "(i) Various games, e.g. chess, noughts and crosses, bridge, poker; (ii) *The learning of languages*; (iii) *Translation of languages*; (iv) Cryptography; (v) Mathematics." [8]. Around the same time, similar ideas were discussed by Warren Weaver and Andrew Booth, suggesting that electronic computers could be used for translating one language into another. In 1949, Weaver distributed a memorandum [9] among potential collaborators that would cause a ripple effect in the research community. For many of its recipients, the ideas presented in the memorandum were revolutionary: They presented possibilities for new and exciting applications of computers, most notably stimulating ideas concerning NLP, which in effect launched machine translation as a scientific enterprise.

The Georgetown-IBM experiment conducted in 1954 is considered a milestone in the early history of NLP as it was the first real public demonstration of machine translation performed on an actual computer, and the first implementation that went beyond word-for-word translation. It involved a fully automatic machine translation of sixty sentences from Russian to English. The experiment and the succeeding research revealed several imperative issues regarding language ambiguity that had to be addressed for the field to progress. Over the next decades, various approaches attempted to overcome these issues. Highlights include Chomsky's establishment of universal grammar with syntactic structures [10] (1957) and Schank's conceptual dependency theory for natural language understanding (1969) [11], which clearly inspired many of the structural analysis methods used today, including part-of-speech tagging and dependency parsing.

In the early history of NLP, writing and executing computer programs were not trivial tasks. It was mainly due to a question of practicability with the punched card machinery, and the ensuing great expenses. As technology progressed, so did the approaches towards natural language processing, and in the late 1980's, the field took a leap when NLP systems employed machine learning algorithms rather than the complex sets of hand-written rules that had dominated the NLP research field since its inception. The shift was made possible by technological advances that led to a significant increase in computational power, which in turn empowered the corpus linguistics that underlies the machine learning approach to NLP. The machine learning approach had up to that point been strongly discouraged by Chomsky's then-dominating linguistics theories. At the time, mainly supervised machine learn-

ing methods were applied, i.e. methods that required a set of labeled data to produce an inferred function that later would be used to classify unlabeled data, including methods such as decision trees and Hidden Markov Models. Eventually, the focus shifted towards statistical and probabilistic language models.

A clear disadvantage of supervised learning methods is that they require large amounts of labeled data, which is very costly to produce. In contrast, semi-supervised learning methods are able to exploit large amounts of unlabeled data with only a fraction of labeled data, while unsupervised learning methods does not utilize labeled data at all. These learning methods are able to exploit less amounts of labeled data than that of fully supervised learning methods, and consequently absorbs fewer resources, at least in that regard. This caught the attention of the NLP research community, and in the subsequent years, NLP research increasingly began to focus on semi-supervised and unsupervised learning methods. In the more recent years, the focus has narrowed down to semi-supervised learning methods. Learning methods in NLP will be further explored in section 2.7.

Although the Georgetown-IBM experiment revealed several limitations in machine translation and NLP research in general, scientists were optimistic in the aftermath of the experiment, claiming that the issues should be resolved within three to five years [12].

Nearly sixty years later, most of the issues still persist, and machine translation has had very limited success, and then almost exclusively in restricted domains such as weather reports. As machine translation at its most basic level is a word-for-word substitution from a source language into a target language, it works well in specific domains with closed environments, and strict, formal wording, which is the case for weather reports. It is however likely to be a challenge in most contexts because the approach assumes an one-to-one correspondence between languages, and because natural language by nature is dynamic, informal, and generally unrestricted in its choice of vocabulary. Because the Georgetown-IBM experiment only included a small set of sentences from two languages, it naturally excluded many aspects of natural language, and over the next decades, it would become apparent that the true complexity and inherent ambiguity of natural language had been immensely underestimated by the scientists, and that the actual extent of natural language went beyond their imagination.

2.2 Language Ambiguity

In most cases, natural language is unrestricted in its choice of vocabulary. A dynamic and image-rich language often use metaphors and idioms, which creates a fertile environment for language ambiguity. This inherent property is what makes machine translation and other natural language applications particularly challenging. In this section, we will describe common concepts in languages, and explain how and why they may introduce language ambiguity.

2.2.1 Collocations and Idioms

Both collocations and idioms are cases where synonyms cannot be interchanged without thorough deliberation, as word-for-word translations may convey entirely different meanings in different languages. **Collocations** are two or more words that often occur together, and where the combination of the words has an additional and often different meaning than the individual words. Consider for example the collocation "fast food". Although "quick" in a grammatical sense is equivalent to the word "fast", "quick food" sounds unnatural for native English speakers. **Idioms** are expressions with figurative, or sometimes literal, meaning. "Barking up the wrong tree" is an example of an idiom that often is used to imply that someone waste their efforts by pursuing the wrong thing or path.

Collocations and idioms are albeit an issue in machine translation, but it is important to emphasize that it also is a challenge in language in general, because they have a figurative sense, and thus have to be interpreted to capture the true meaning of the expressions.

2.2.2 Words with Multiple Meanings

Certain words and expressions may have different meanings, depending on their context. For instance, to "withdraw" in a financial context means to take money out of the bank, while in a game of poker, it means that the player quits the game. Other examples of words with multiple meanings include homonymy and metonymy. **Homonyms** are words that are spelled the same way, but that have at least two distinct and unrelated meanings. For example, a "bank" could refer to a river bank or a financial institution. **Metonymy** is when an entity is referred to by the name of an associated entity, rather than its own name. For example, it is common practice to refer to a country's capital city when referring to a country's government, as demonstrated in example 2.1.

- (2.1) *Moscow* is yet to disclose at what level it will be represented at The Hague meeting.

It is not unlikely that "Moscow" would have been labeled as Russia's capital city, but the real entity in play is actually the Russian government, which can be deduced from a city's versus a government's ability to disclose anything.

2.2.3 Syntactic Ambiguity

In English, the correct order of words depend on their grammatical category, i.e. whether they are nouns, prepositions, verbs, adjectives, and so on. The words are organized into so-called **phrases**, which are groupings of words that acts as constituents of a sentence. **Constituents** serves as single units within the hierarchical structure of a sentence, and are characterized by their ability to be placed in various positions in a sentence without altering the meaning of the sentence. This also includes uniform syntactic possibilities of phrase expansion and phrase reduction [13].

- (2.2) Clark got sick.
(2.3) The teacher got sick.
(2.4) The tall teacher with the curly hair got sick.

To understand what this entails, let us consider the sentences in the above examples. The phrases "Clark", "The teacher" and "The tall teacher with the curly hair" may look quite different, but they could indeed convey the same thing, given that they refer to the same person. Moreover, the phrase in example 2.4 could also be reduced to that of the sentence in example 2.3, and likewise, the sentence in example 2.3 could be expanded to that of the sentence in example 2.4.

- (2.5) The children ate the cake with a spoon.

Another type of syntactic ambiguity is **attachment ambiguity**, which occur when a phrase could have been generated by two different root nodes. In example 2.5, the sentence could either mean that the children used a spoon to eat the cake, or that the children ate a specific cake that had a spoon in it, and not say strawberries or icing, depending on whether the preposition is attached to "ate" or "cake".

These are some of the examples of syntactic ambiguity that illustrates how am-

biguity caused by syntax may impact the semantic interpretation of the meaning of phrases.

2.2.4 Phrasal Scope

A particular subject can have a scope that extends over one or more sentences or paragraphs. The sentence in example 2.6 can for instance be interpreted as that nobody liked the food, or that a least one person did not like the food. To deduct the actual meaning of the sentence, we must determine which of the interpretations is correct given the context.

(2.6) Everyone didn't like the food

There could also be covert relationships between sentences within a scope, often referred to as a **discourse**. A central problem in discourse analysis is resolving anaphoric relations., which are when noun phrases refers to the same person or thing. An example of anaphoric relation resolution is demonstrated in example 2.7, where *Barack Obama* and *He* refers to the same person.

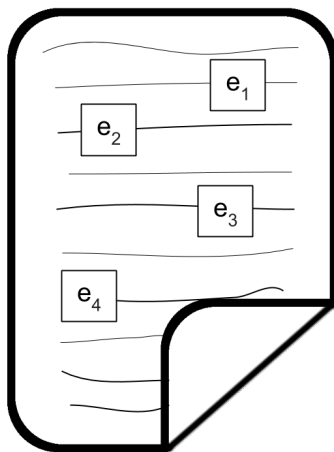
(2.7) *Barack Obama* was elected president of the United States in 2009. *He* and his family resides in the White House.

Assuming our knowledge only confines to the information given in the example above, it cannot easily be deduced that the White House referred to is not just an ordinary house painted white, but in fact is a particular location in Washington D.C in United States where the inaugurated president and his family resides. This information facts may however be possible to extract if the entities in the text is connected to a knowledge base that discloses these facts.

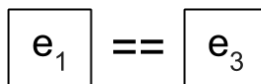
2.3 Ambiguous Named Entities

A **named entity** is a real-world object, such as a person, location, an animal, or a time period, and an **entity mention** in a text is a reference to a particular entity. In example 2.7, *Barack Obama*, *United States* and *White House* are all examples of named entities. Entities are usually identified by their proper name, but unfortunately there seldom is a one-to-one correspondence in the mapping between entities and names. This many-to-many relationship is caused by two different phenomena:

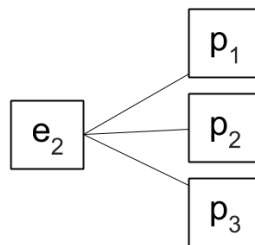
1. **Synonymy**, where the same entity is referred to by several names
2. **Polysemy**, where a mention in a text could denote multiple entities.



(a) Document with four entity mentions



(b) Synonymy



(c) Polysemy

Figure 2.1: Synonymy and polysemy

Figure 2.1 illustrate these phenomena on a conceptual level. The entity mentions are denoted $e_1 \dots e_n$, and $p_1 \dots p_n$ are entity pages. If two or more entity mentions refer to the same entity, they exemplify synonymy, illustrated in figure 2.1 (a). If an entity mention could be linked to more than one entity, it exemplifies polysemy, shown in figure 2.1 (b).

2.4 Entity Linking

Entity Linking (EL) is the task of disambiguating ambiguous entity mentions by linking ambiguous entity mentions to actual world entities [14]. EL is typically split into two subtasks: **Name Entity Recognition (NER)** and **Name Entity Disambiguation (NED)**. NER refers to identifying entities in a text and classifying them into predefined categories (see section 2.5.3), while NED establish mappings between ambiguous entity mentions and their corresponding entities in a knowledge base. In some ways, EL is similar to coreference resolution, but rather than clustering entity mentions in a single document, it spans over multiple documents, and links to specific entities in a knowledge base. An EL system has to cope with three main issues: Cases of missing entities, synonymy, and polysemy.

2.4.1 Missing Entities

Many EL systems assume that the target entity always is present in the knowledge base. However, there is a chance that the text corpus contain entity mentions that refer to entities that are not present in the knowledge base. It could be because the entity is referred to by a name variation, misspelling, or a nickname, or because the knowledge base is not up-to-date. It is also a possibility that it is unclear which entity that is being referred to in the text.

2.4.2 Synonymy

The synonymy aspect of named entity ambiguity can in part be solved by considering entity-specific values present in a knowledge base, e.g. an entity's name, birth name, aliases and nicknames, and combining these values with techniques such as query expansion and spelling correction. This aspect is however not a focus of this thesis.

2.4.3 Polysemy

Various approaches have been proposed for solving the polysemy aspect of named entity disambiguation. It includes considering entity coherence and semantic similarity between two Wikipedia pages by comparing their set of incoming and outgoing links [15], collectively assigning entity mentions to entities based on the assumed relationship between the entity mentions [16], and by comparing the context of an entity mention to the contextual information about potential target entities [14]. Section 2.8 will revisit some of these approaches and elaborate further on some key systems that applies entity linking for named entity disambiguation.

2.5 Information Extraction

The following sections introduce information extraction techniques that may be used to identify entities in natural language text. As NLP is a quite comprehensive research field, we will only focus the main concepts relevant to this thesis. More specifically, we will describe the annotators that are provided by the Stanford CoreNLP parser that may be used to identify and disambiguate ambiguous entities. We adhere to the standards set by the Stanford CoreNLP parser because its output format is supported by the information extraction framework DeepDive, which is an information extraction framework that will be used to conduct our experiments.

2.5.1 Part-Of-Speech

Part-Of-Speech (POS) are grammatical categories of terms that share similar syntactic behaviour, such as nouns, verbs and adjectives. In linguistics, **part-of-speech tagging (POS tagging)** is the process of marking up terms in a corpus to its corresponding part-of-speech. The process is based on both the word's definition, as well as its surrounding context, i.e. the word's relationship with adjacent and related words in a phrase.

The selection of grammatical categories may vary among POS taggers, depending on which tag sets the POS tagger uses. The tag sets differ in annotation and granularity, e.g. they often refer to grammatical categories in different "codes", and some tag sets may distinguish between verb tenses, while other does not. Tag sets generally incorporates morphological distinctions of a particular language and are thus not directly applicable to other languages. Brown, C5 and Penn Treebank are examples of English tag sets. The POS tagger used in the Stanford tool suite uses the standards defined by the Penn Treebank tag set, and is based on the log-linear POS taggers described in [17]. We will use the terminology as specified by the Penn Treebank tag set when referring to POS tags for the remainder of this thesis.

To illustrate how POS tags are used in practice, we use the first sentence from example 2.7¹, resulting in:

Barack	Obama	was	elected	president	of	the	United	States	in	2009	.
NNP	NNP	VBD	VBN	NN	IN	DT	NNP	NNPS	IN	CD	.

¹Parsed with the online Stanford Parser available at [HTTP://nlp.stanford.edu:8080/parser/](http://nlp.stanford.edu:8080/parser/)

The `NNP` and `NNPS` POS tags are used to mark proper nouns². In combination with structural analysis, proper nouns can be exploited to identify named entities in text as well as to determine which entity that is most likely to be the main topic of a document.

2.5.2 Phrase Structure and Dependency Parsing

There are several ways to represent the structure of a sentence, generally categorized by two types of parsing: **Phrase structure parsing**, which focus on identifying phrases and their recursive structure, and **dependency parsing**, which focus on the relations between individual words.

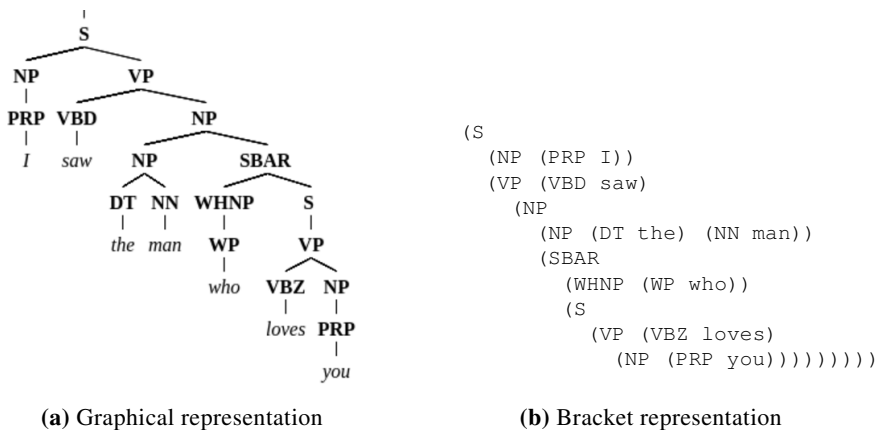


Figure 2.2: Different representations of a constituent parse tree for the sentence "I saw the man who loves you".

In its simplest form, phrase structures could be represented as a constituent parse tree, which is simply a nesting of phrase constituents. Figure 2.2 illustrate different representations of a constituent parse tree. The Penn Treebank distinguish constituents on three levels: Clause level, phrase level and word level³. This information

²An overview of standard English POS tags and their expanded forms used in Penn Treebank can be found at ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

³Documentation available at surdeanu.info/mihai/teaching/ista555-fall113/readings/PennTreebankConstituents.html

can be used to detect entities, e.g. by identifying the noun phrase of the parse tree on the phrase level (denoted NP in figure 2.2).

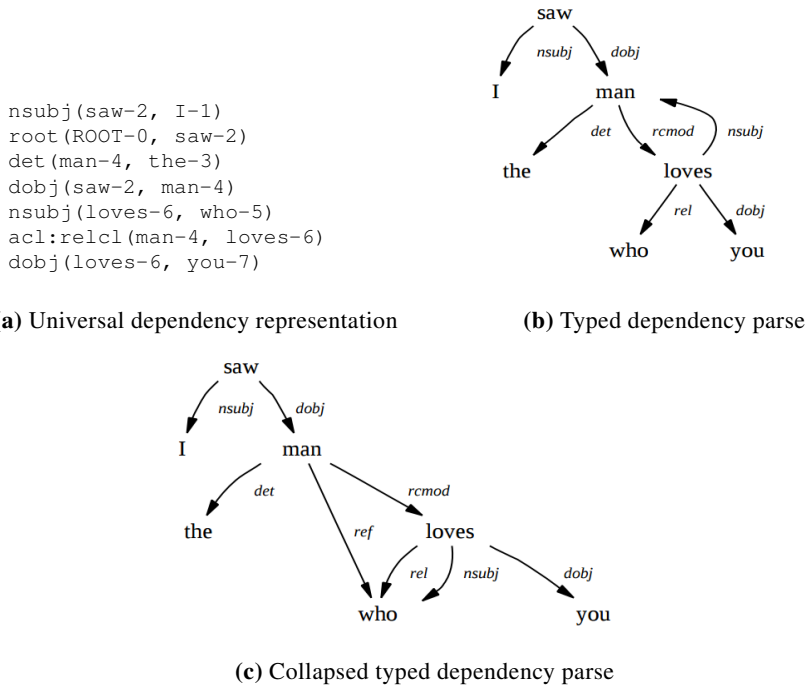


Figure 2.3: Different representations of a dependency parse for the sentence "I saw the man who loves you". Graphs reprinted from [2]

While basic dependency parsing simply establish relationships between words, there are other variations of dependency parsing that provides more information. For example, a **typed dependency parse** additionally labels dependencies with grammatical relations [2], while in a **collapsed typed dependency parse**, grammatical components such as prepositions and conjuncts are collapsed in order to establish more direct dependency links between important content words, which could be useful in information extraction applications, e.g. for pattern simplification [18]. Figure 2.3 displays different variations and representations of a typed dependency parse of the recurrent example sentence "I saw the man who loves you".

2.5.3 Named Entity Recognition

Named Entity Recognition (NER) is the task of identifying mentions of named entities in text and classifying them in predefined categories. There is no standard way to categorize named entities, and as a result there are often discrepancies between the different parsers in terms of their selection in named entity categories. We adhere to the standards set by the Stanford CoreNLP parser.

Type	Tag	Description	Example
Named	PERSON	Person	"Costigan", "Michelle Obama"
	LOCATION	Location	"United States", "Trondheim"
	ORGANIZATION	Capitalized proper name	"Time magazine", "Tupperware"
	MISC	Miscellaneous entities	"Studio", "Galaxy", "Democratic"
Temporal	TIME	Time expression	"12 o'clock", "night"
	DATE	Date expression	"Dec. 19", "Monday", "recently"
	DURATION	Duration	"Hour", "Week"
	SET	Set of times, e.g. time intervals and recurring events	"Weekly", "Mondays"
Numerical	MONEY	Monetary values	"\$79", "millions of dollars"
	NUMBER	Numeric value	"2", "two"
	ORDINAL	A word representing the rank of a number.	"first", "second"
	PERCENT	Digit attached to percentage character.	"50%"
Other	O	Miscellaneous words not recognized as named entities	"wrote", "them", "well-known"

Table 2.1: Overview of named entity categories

According to its website⁴, the Named Entity Recognizer in the Stanford CoreNLP parser, henceforth known as Stanford NER, recognize named, numerical, and temporal entities. The named entities are identified using a combination of three CRF sequence taggers [19]⁵, numerical entities are identified using a rule-based system

⁴stanfordnlp.github.io/CoreNLP/ner.html

⁵See nlp.stanford.edu/software/CRF-NER.shtml for more information.

with regular expressions, while temporal entities are identified by a rule-based temporal expression recognizer, SUTime [20]⁶, currently only available in English. An overview of these named entity categories are given in table 2.1

2.6 Temporal Information Extraction

Temporal information extraction is valuable in research areas such as question-answering, information extraction and summary. This section will elaborate on different types of temporal data, as well as the two main categories of temporal data, namely *content-based temporal data* and *non-content-based temporal data*.

2.6.1 Types of Temporal Data

A document or an entity is often associated with some type of temporal data, of which the most pertinent type are *temporal expressions*, which may be categorized as explicit, implicit or relative [21].

Explicit temporal expressions can be mapped directly to an exact time point or time interval, and can thus be normalized without any additional information, e.g. "17th May, 2016" and "Week 33, 2015". **Implicit temporal expressions** are represented as imprecise time points or time intervals, e.g. "New Year's Eve 2015" and "Spring of 1945". These implicit temporal expressions can be mapped to their corresponding explicit expression. For example, the implicit temporal expression "New Year's Eve 2015" can be mapped to the explicit temporal expression "31th December, 2015".

(2.8) This week, president Bush announced that . . .

Relative temporal expressions is when the content of the document is relative to either the explicit or implicit temporal information in the document, or the document's metadata. In example 2.8, "This week" refers to a point of time that is relative to when the document was published. The time of the publication could potentially help distinguish between the two ambiguous entities named president Bush, George Bush Jr. and his father George Bush Sr., for example if the document was published before George Bush Jr. became president. Figure 2.4 exemplifies the kind of context information that is needed for normalization of relative temporal expressions. Here, the relative temporal expressions "the following year", "today" and "December" are

⁶Available at nlp.stanford.edu/software/sutime.shtml

Document Creation Time: 1998-04-18
 Hungarian astronaut Bertalan Farkas is leaving for the
 United States to start a new career, he said today .
 ... On May 22, 1995, Farkas was made a brigadier general,
 and the following year he was appointed military attache
 ... However, cited by District of Columbia traffic police in
 December for driving under the influence of ...

Figure 2.4: Examples of explicit (transparent boxes) and relative (solid boxes) temporal expressions. Arrows indicate what kind of context information is needed to normalize the temporal expression. Reprinted from [3].

linked to explicit temporal expressions in the document content and the document's metadata, respectively.

2.6.2 Content-Based Temporal Data

As the name suggests, content-based temporal data considers the textual content of documents. Two of the most interesting aspects that can be drawn from temporal expressions in the content is the time period that is the focus of a document, *document focus time*, and how entities evolve over time, *entity evolution*.

Document Focus Time

Document focus time is the period of time that is referred to in the document contents. The document focus time is generally determined by extracting temporal expressions from the text. This extraction process is normally conducted by 1) identifying time entities in the text, and 2) performing temporal reference resolution in order to normalize the relative temporal expressions into explicit or implicit temporal expressions [22]. The time entities is typically one of four temporal types: date, time, duration or set [23]. The date and time types refer to specific points in time, e.g. "Four o'clock" or "17th May, 2016". In contrast, the duration type conveys information about the length of an interval, e.g., "one week", while the set type inform about the periodical aspect of an event, e.g., "once a year".

Resolving temporal expressions with respect to a reference date is typically rule-based. For example, the temporal tagger used in the Stanford CoreNLP parser, SU-Time [20], is a deterministic rule-based system built on regular expressions patterns according to the TimeML annotation language [24]. Other studies use similar approaches, such as GUTime [25]⁷, which is a temporal tagger that use a blend of hand-crafted and machine-discovered rules.

Entity Evolution

It has been observed that entities often change over time, which causes ripple effects in both how the entities evolve and how they are represented. This evolution can be manifested in two problems, namely *terminology evolution* and *context change* [22]. **Terminology evolution** includes changes of words related to their definitions, semantics and names. For example, there may be a difference in spelling between modern and historic language, new words or spelling variations may be introduced, and the meaning of a word may change. Typical **context changes** includes changes in personal relationships, e.g. name change upon marriage or divorce, world-related knowledge, and organizational roles.

In short, exploiting temporal expressions in content-based temporal data makes it possible to track specific people, events, organizations and other entities, draw timelines and anticipate changes that are likely to occur during specific phases of life in general, and also in organizational and societal settings. Furthermore, it can follow how languages progresses, as changes in language is a continuous process that is observable even in short periods of time [22].

2.6.3 Non-Content-Based Temporal Data

The temporal data that does not involve the content itself, but rather the properties of the document, are often referred to as non-content-based temporal data. Time of creation and publication are typical examples of non-content-based data, and are often needed during resolution of temporal expressions that are relative to the publication time, or document creation time, as illustrated in figure 2.4 on page 20. These values are usually represented as timestamps and assigned to documents. Disadvantages for methods that make use of non-content-based temporal data is that they heavily depend on these values, but does not take into account that this type of information not always is available, nor accurate or trustworthy, because the metadata often only

⁷urltimeml.org/tarsqi/modules/gutime/index.html

specifies when the document lastly was modified or crawled [26]. This put strong constraints on the availability and accuracy of other information, which could make non-content-based temporal data less useful in practice. Despite of this, several tasks use non-content-based temporal data in their research, such as time-aware search and temporal clustering [3].

2.7 Learning Methods

Learning methods directed at the named entity disambiguation problem generally fall into three types of learning methods: the unsupervised approach, the supervised approach, and the semi-supervised approach.

2.7.1 Unsupervised Learning Methods

In unsupervised learning methods, all the input data is unlabeled, and the aim is to detect and describe the hidden structure of the data. Unsupervised learning methods mainly rely on heuristic rules or predefined similarity metrics. They are simple and easy to implement, but usually produce poorer results than supervised methods [27]. This is because the unsupervised methods usually does not focus on any particular relations or features, and thus may extract large numbers of noisy and inaccurate relations, which may not be easy to map to corresponding relations in a knowledge base.

2.7.2 Supervised Learning Methods

Supervised learning methods requires both training data and test data as input. The training data consist of a set of labeled examples that each represents a pair that consist of an input object and desired output value. The method analyzes the training data and produces an inferred function, which can be used for classification of new and unlabeled examples from the test data. This approach suffer from two important problems:

1. **There often are practical challenges in creating or obtaining enough training data.** It is resource consuming to produce labeled training data, which consequently limits the quantity. Additionally, it may be difficult to obtain training data that suits the applications of the system. For example, you may have access to a semantically tagged dataset that contains Twitter data from 2011 to 2013, but may be interested in social media in general, or another time interval.

2. **Supervised approaches are especially prone to overfitting.** This is because the examples in the training data are constructed on a corpus, which produces classifiers that may be biased toward that particular corpus. Consequently, this may result in poor accuracy when the trained model is used on other test data.

2.7.3 Semi-Supervised Learning Methods

The semi-supervised approach is a sort of hybrid of the two aforementioned learning approaches. It uses a small amount of labeled data with a large proportion of unlabeled data. There are mainly two ways to generate labeled data in a semi-supervised learning method: 1) Manual labeling, in the same manner as supervised learning methods, and 2) Distant supervision, which either rely on heuristic rules or mappings from a secondary dataset.

Distant Supervision

In distant supervision with mappings from a secondary dataset, the idea is to extract specialized datasets from some source of data, for example a knowledge base using SPARQL queries, and use the resulting dataset as training data to classify entities. For example, one can use SPARQL queries to extract all entities belonging to a particular Wikipedia category, or that are affiliated through a particular relationship, e.g. married couples, children and parents, siblings, or the literary works of a particular author. One could also use heuristic rules to define entity properties, or relations between entities, but in this thesis we apply a semi-supervised learning method with distant supervision that extracts training data from a knowledge base and use it to extract information about entities in a semantically and syntactically tagged text corpus.

2.8 Related work

Named entity recognition and disambiguation in natural text is a key component in many research fields. In this section, we will briefly review related work that touches upon the named entity disambiguation problem.

2.8.1 Entity-Centric Knowledge

Over the past twenty years, extraction of entities has been of particular interest in the information retrieval research community, resulting in IR evaluations such as the Message Understanding Conference (MUC), Automatic Content Extraction (ACE), Text Analysis Conference (TAC), and Text REtrieval Conference (TREC)⁸. All of these evaluations⁹ focus on NER in various *tracks*, i.e. areas of focus in which particular IR or NLP tasks are defined. Of the aforementioned conferences, TREC and TAC are the only conferences that are still operating. Both conferences are organized by National Institute of Standards and Technology (NIST), and have had several interesting tracks over the years, but for the focus area of this thesis, there are two tracks that stand out, namely **Knowledge Base Population (KBP)** and **Knowledge Base Acceleration (KBA)**, which both are entity-centric, and utilize a semantically tagged stream corpus as a mean to expand and update knowledge bases. The tasks of these conferences changes every year, and as the information from 2015 either was incomplete or unavailable at the time of writing, this thesis is based on the conferences' information and results from 2014, although we endeavour to keep the information as up-to-date as practicable.

Knowledge Base Population

TAC introduced a dedicated KBP track in 2009. The ultimate goal of TAC KBP is to develop and evaluate technologies for building and populating knowledge bases. This is done by inputting a knowledge base and a large unstructured text corpus, and extracting useful information from the corpus to supplement the incomplete elements of the knowledge base, or to update existing elements in the event of context changes. The perhaps most interesting tasks of KBP are Entity Linking (EL) and Slot Filling (SF). EL follows the same definitions as specified in section 2.4, while the SF task is to search a corpus for new information to update or fill in values for predefined slots for a given entity in a knowledge base. A predefined slot could for example be birth date, or other temporal information, as well as explicit information

⁸trec.nist.gov

⁹NER was added as a task at the sixth MUC conference in 1995.

such as birth place, occupation, or spouse. Combining slot filling and entity linking could thus be an interesting approach towards named entity disambiguation, also in a temporal aspect.

In the TAC KBP competition of 2014, Stanford University's system [5] achieved the highest overall score with respect to all the evaluation metrics [7]. In their submission, Stanford used the information extraction framework DeepDive¹⁰. DeepDive use distant supervision to extract training data from a secondary dataset, and use combine this training data with rule-based approaches to extract information from input text [28]. The design of Stanford University's contribution served as a baseline for our experiment design, differing in that our efforts are directed towards whether it is possible to exploit temporal data to improve named entity disambiguation.

Knowledge Base Acceleration

The KBA track was initiated by TREC in 2012 inspired by ideas from TAC KBP, and in response to the challenges faced by content managers concerning the exponential growth of online data and its effect on the relevance and correctness of information in knowledge bases. According to its website¹¹, the TREC KBA track ran as an open evaluation in TREC from 2012 to 2014, and was succeeded by TREC Dynamic Domain in 2015.

The ultimate goal of the TREC KBA evaluation track was to develop systems to aid construction and maintenance of knowledge bases by automatically recommending edits based on content from a time-ordered stream corpus [29]. An important task in KBA is Cumulative Citation Recommendation (CCR)¹², which is to use this stream corpus in conjunction with a predefined set of entities from a knowledge base, and generate a score based on the pertinence of a document considering a particular entity from the entity set. Stream Slot Filling (SSF) exploits the nature of stream corpora to follow target entities as they evolve over time to provide updated slot values about particular entities within the timeframe provided by the corpus scope.

The temporal focus of the KBA track provides several possibilities regarding extraction of temporal slot values. Balog et. al suggests to capture entity-related changes by considering time bursts through Wikipedia page view statistics, and refers to implicit handling of named entity disambiguation through centrality detection [30].

¹⁰deepdive.stanford.edu

¹¹trec-kba.org

¹²Also known as Vital Filtering as of TREC KBA 2014

Abbes et al suggest to leverage the content-based temporal expressions of a document to determine the relevancy of the document. Although the approach was deemed useful for detecting vital documents, i.e. documents containing timely relevant information about an entity, it assume that all entities in the evaluation set is present in Wikipedia, and does not consider the named disambiguation problem [31].

2.8.2 Spatiotemporal Entity Linking on Microblogs

To the best of our knowledge, the only research effort that has been made into temporal entity linking to this day is a study of temporal entity linking on microblogs in 2014 [32]. The study suggests that entity linking is an essential step for applications that exploits user data affiliated with microblogging services, and proposes to prove its potential by performing entity linking on Tweets¹³. Due to Twitter's 140-character limitation, Tweets are short, and usually noisy and colloquial by nature, which makes it especially challenging to conduct entity linking. It is also important to note that all Tweets are linked to a timestamp, and that they also have the possibility of being mapped to a location. These facts puts Tweets in a fortunate position because it creates a semi-structured dataset that differs from traditional raw text in that its temporal data is normalized, attached and easily accessible for all Tweets.

The study use a database, containing a set of entities that a tweet can link to, and a lexicon, based on information from disambiguation pages, redirect pages and anchor texts in Wikipedia, that maps candidate anchors to a potential entity set in the database. The study concludes with that spatiotemporal signals are crucial for entity linking on microblogs. This claim is supported by comparing the recall, precision and F_1 -score results from similar work, which demonstrates that although the study did not have any significant impact on precision, it did improve recall and F_1 -score.

2.8.3 Wikification

Wikification is a specialization of entity linking where Wikipedia serves as the target knowledge base. Over the recent years, several studies have utilized Wikipedia, either as a reference knowledge base, or by exploiting semi-structured and structured data embodied in Wikipedia. Wikipedia's link structure facilitates for many possible approaches. For example, the dataset of which Wikification has been applied spans from Tweets [33, 34], snippets of search-engine results [33], news articles [15, 35],

¹³A Tweet is a posting made on the social media website Twitter. It may contain photos, videos, links and up to 140 characters of text.

and Wikipedia articles [15]. Training data for disambiguation purposes has been generated in various ways, including by exploiting Wikipedia’s hyperlinks to create a dataset of ambiguous queries [36], keyword extraction [37], category labels extracted from the first sentence of a Wikipedia article [38], the relatedness between unambiguous and ambiguous entity mentions [15], and Wikipedia statistics in conjunction with semantic and syntactic tagging of named entities [35].

Entity linking research the recent years has been greatly influenced by the the ideas presented in [15], which can be condensed into the following key points: (1) identifying a set C of *context pages*, which are pages linked to by unambiguous entity mentions, (2) a relatedness between two Wikipedia pages, p_1 and p_2 , that is based on the overlap between the pages that links to p_1 and p_2 , and (3) the notion of coherence of a page between other pages in set C . These concepts were adapted and further developed in [16], which suggested to jointly consider all entity mentions in an input, and aim for a collective entity linking with respect to coherence between the potential target entities. The similarity between potential target entities should then be measured with context similarity score between each pair of entity mention and corresponding potential target entity.

One of the most notable entity linking systems based on the aforementioned ideas are TAGME [33]¹⁴, which is a system that enhances plain text with pertinent hyperlinks to Wikipedia pages. TAGME specialize in annotating short and noisy data, such as snippets of search-engine results and tweets, and aims to identify possible entity mentions in input text and link these mentions to unambiguous entities extracted from Wikipedia. Although TAGME explicitly targets short text, it also yields competitive results in annotation of longer texts, which makes TAGME an interesting candidate with respect to an entity linking experiment. However, as argued by [39], the full set of TAGME experiments described in [33] are not repeatable due to inter alia discrepancies in the datasets. Nonetheless, TAGME is an outstanding entity linking system, and will therefore be used for initial experiments to test the quality of our entity linking.

¹⁴Available at tagme.di.unipi.it

This chapter presents an overview of the system flow, describes the details of the system components, and justifies the technical and practical decisions made during development.

3.1 System Flow

The main goal of this thesis is to investigate whether it is possible to improve named entity disambiguation by considering the temporal aspect of the data. Figure 3.1 depicts a high-level description of how such a system could be constructed. The system flow is divided into the following steps:

1. Obtaining the dataset
2. Preprocessing data
 - (a) Restraining the data
 - (b) Preparing data for parsing
 - (c) Parsing data
3. Distant supervision
 - (a) Extracting relationships from the data sources
 - (b) Extracting and downloading data from knowledge base

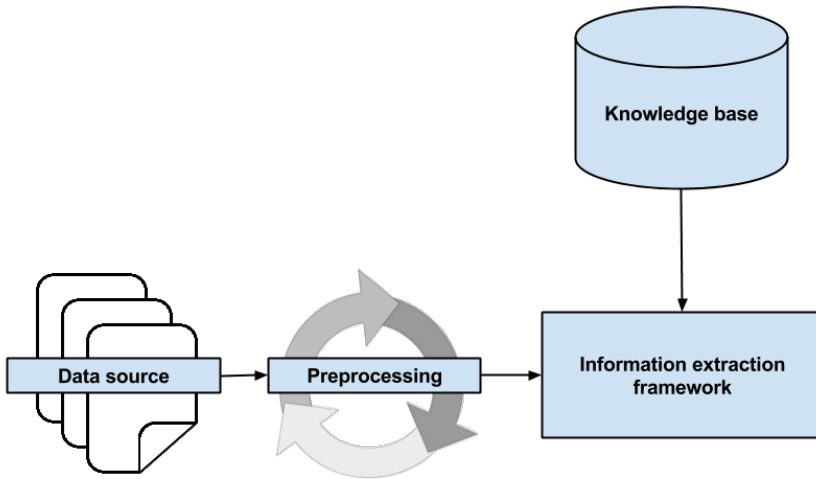


Figure 3.1: High-level description of system

3.2 Datasets

In this section, we describe the contents and structure of the datasets that may be used in our system. This include the components referred to as "Data source" and "Knowledge base" in figure 3.1.

3.2.1 Data Source

As the experiment results will depend on the dataset of which it is constructed, it is important to have a representative dataset. Ideally, the dataset is a semantically tagged document corpus that contains documents from several heterogeneous data sources, and is cleansed for characters with a special meaning that may cause problems in a processing context, e.g. when loaded into databases or during parsing.

3.2.2 Knowledge Base

A knowledge base represents facts about the world or particular domains according to a model that defines the vocabulary for a domain of knowledge, and the relations between concepts in the domain, also known as an **ontology**. Knowledge bases are

frequently used in applications that acquire a reference point for established and recognized knowledge. To select a knowledge base that befits our needs, we present the following requirements:

- **Open and freely available** Due to constraints in time and finances, the KB should be freely available and easy to access.
- **Up-to-date and broad coverage of general knowledge** To ensure a good correspondence between the entities in the corpora and the KB, the KB should be up-to-date and have a broad coverage of general knowledge. It may preferably be based on Wikipedia, as Wikipedia is a good data source because it is relatively up-to-date, and because the content of the articles tend to make explicit many facts that might be omitted in newswire.
- **Incorporate Semantic Web standards** To enable an investigation of semantic and syntactical properties of the corpus, the KB should embody Semantic Web standards such as data modelling with RDF and querying with SPARQL.

These requirements excludes small or weakly structured knowledge collections, discontinued projects such as Freebase, and commercial projects with no or limited access such as the Google Knowledge Vault [40], Walmart [41], Facebook [42], WolframAlpha, Internet Movie Data Base (IMDB), and Microsoft's Satori. It also excludes Wikipedia, which does not facilitate semantic querying of its data.

The strongest candidates are **DBpedia**¹, **YAGO** [43], and **Wikidata**². They all store their data as Resource Description Framework (RDF) tuples and can be accessed through the query language SPARQL. Additionally, they cover general knowledge and gather at least some of their data from Wikipedia.

There is however a great difference that distinguishes Wikidata from the other two KBs: Rather than extracting information from Wikipedia, Wikidata aims to *provide* information, most notably by automatically creating, updating and enriching parts of Wikipedia articles. As our objective is to extract information, this fact disqualifies Wikidata from our candidate selection, and thus moves our attention to the two remaining knowledge bases.

As demonstrated in table 3.1, both DBpedia and YAGO extract an ontological knowledge base from Wikipedia, they both are registered in Linked Open Data (LOD)³,

¹ wiki.dbpedia.org

² wikidata.org

³ The LOD cloud is a web of interrelated open datasets, available at urlrllod-cloud.net

	YAGO	DBpedia
Fact representation	Subject-predicate-object triple with time and location (SPOTL)	Subject-predicate-object (SPO) triple
Release date	2008	23rd January, 2007
Data source	Wikipedia, WordNet, GeoNames, Wikidata	Wikipedia
Influence on other datasets	SUMO, DBpedia, UMBEL, Freebase	Freebase, YAGO
LOD linkage	DBpedia	Freebase, OpenCyc, YAGO, UMBEL, GeoNames, Musicbrainz, CIA World Factbook

Table 3.1: Overview of KB features

and they use subject-predicate-object triples to represent facts. When it comes to coverage, however, DBpedia is deemed as the most prominent KB in the LOD cloud, and often referred to as the LOD hub [44, 45, 46], while YAGO is only connected to the LOD cloud via DBpedia. DBpedia's advantage is partially outweighed by the fact that YAGO retrieves its data from several data sources, while DBpedia only extracts data from Wikipedia. Furthermore, YAGO holds an advantage because time and location information is attached to facts by reification.

It was primarily three reasons that ultimately made us choose DBpedia over YAGO: DBpedia's close integration with Wikipedia, easier access, and because it offers the highest number of datatypes⁴, which we have assumed entails a higher granularity, and thus potentially could offer more temporal information. As DBpedia receives all its data from Wikipedia, it entails that

1. it inherits the broad coverage of general knowledge from Wikipedia,
2. it can more easily exploit the link structure in Wikipedia, and
3. potential problems caused by different setup or standards in various data sources is avoided.

Additionally, DBpedia provides a simple web interface for online SPARQL queries⁵, and easily accessible data dumps of Wikipedia-specific categories⁶. The next sec-

⁴mappings.dbpedia.org/index.php/DBpedia_Datatypes

⁵dbpedia.org/sparql

⁶wiki.dbpedia.org/Downloads2015-10

tions will describe the semi-structured data that DBpedia extracts from Wikipedia [44], and that may be exploited to cope with the named entity disambiguation problem.

Basic Wikipedia Article Structure

The title of a Wikipedia article, or *entity page*, acts as a unique identifier for the entity page. The title suggests what the article is about, and consists of a sequence of words, separated by underscores, with the first word capitalized. The title of the article is usually the entity's most commonly used name, to ease the entity recognition and search for ordinary users [36, 47].

In cases of polysemy, the convention in Wikipedia is to add distinguishing information, often a descriptive parenthetical expression after the article title [47]. For example, to distinguish Franz Ferdinand, the Scottish band, from the archduke of Austria, the entities are denoted "Franz_Ferdinand_(band)" and "Franz_Ferdinand", respectively.

Wikipedia Redirect Pages

A redirect page is an empty page that acts as a pointer to another page, or a section of a page. The purpose of redirect pages is to handle synonymy, i.e. different name varieties of the same entity. Some of these varieties includes alternative names or spellings, common misspellings, closely related words and abbreviations [36, 48].

Wikipedia Disambiguation Pages

A disambiguation page is created in the case of polysemy, i.e. when two or more entities share the same name. Such a page contains a list of references that links to pages for entities that are most commonly known by a particular entity name, and is denoted "<entity-name>_(disambiguation)".

Wikipedia Categories

According to Wikipedia guidelines⁷, every article in Wikipedia should belong to at least one category. These categories usually are conceptual defining characteristics of the article's main subject. Categorization of articles allows the entities to be associated with one or more topics, which in turn could be further categorized by associating them with one or more parent categories. For example, a document

⁷en.wikipedia.org/wiki/Wikipedia:Categoryization#Articles

containing the mentions "Sagan" and "Tyson" is likely to refer to the American astrophysicists Carl Sagan and Neil deGrasse Tyson because they share six common categories: "Science communicators", "American skeptics", "Space advocates", "Planetary scientists", "Fellows of the Committee for Skeptical Inquiry" and "People associated with the American Museum of Natural History". In contrast, other entity candidates for these entity mentions, such as the Slovak professional road bicycle racer Peter Sagan and American boxer Mike Tyson, does not share any common categories.

Wikipedia Hyperlinks

Hyperlinks can be thought of as branches that connects the leaves, i.e. the article pages, to the Wikipedia knowledge tree. When an article contains mentions of entities that already have a corresponding article page, the convention is to link at least the first mention to the corresponding article page by using links or piped links. The article page that contains the hyperlink is called an *anchor*, while the article page that the hyperlink points to is called a *target*.

- (3.1) Oslo is the capital and the [[List of towns and cities in Norway|most populous city]] in [[Norway]].

An example of regular links and piped links are displayed in example 3.1, retrieved from the wiki source code of a sentence from the Wikipedia article on Oslo. The string from the second link, "Norway", denotes the title of the target article. If the article author should wish to display something else than the title in the text, e.g. "most populous city" instead of "List of towns and cities in Norway", then the alternative string is included after the title string in a piped link, as demonstrated in the first link in example 3.1. Consequently, the display string for the aforementioned example become: "Oslo is the capital and the most populous city in Norway."

Wikipedia Infoboxes

The purpose of a Wikipedia info box is to summarize key facts that appears in its associated article page. It is useful because it transforms the unstructured content of the article page into a machine-readable format that can be exploited by third-parties such as DBpedia. Wikipedia has several templates for Infoboxes that is specialized for particular entity types, such as people and companies.

We focus on the template for person entities because they are subject to frequent context changes, such as birth, marriage, education and death, and consequently is

more likely to be affiliated with more temporal data than other entity types. They are also the entity type with the largest number of instances: According to SPARQL queries to DBpedia, DBpedia contain 1.760.735 distinct person instances, in contrast to 83.220 distinct company instances⁸. Although there may not be the exact representation of the number present in Wikipedia, we expect the ratio to be of similar proportions.

Note however that according to Wikipedia guidelines⁹, there are no fixed rules when it comes to usage of infoboxes, or parts of the infoboxes, in Wikipedia articles. This is determined for each individual article through discussion and consensus between the editors. It is also worth noting that the information DBpedia extracts from Wikipedia infoboxes is not cleaned nor merged beyond a minimal amount of property value clean-up, e.g. date normalization. Combined with the fact that Wikipedia does not have any fixed rules when it comes to usage of infoboxes, this creates an environment for noisy and replicate data. In fact, DBpedia state that the infobox dataset they provide only should be used if you require "complete coverage of all Wikipedia properties . . ." and are prepared to accept ". . . relatively noisy data"¹⁰.

Available Temporal Data

The temporal parameters on person entities available in the person infobox template is summarized in table 3.2¹¹.

⁸Results available at bit.ly/1OSOAdP and bit.ly/1WKIhf0

⁹en.wikipedia.org/wiki/Help:Infobox

¹⁰wiki.dbpedia.org/services-resources/datasets/dbpedia-datasets

¹¹Retrieved and adapted from en.wikipedia.org/wiki/Template:Infobox_person

Parameter	Explanation
birth_date	Date of birth: birth date and age (if living) or birth date (if dead).
baptised	Date of baptism: Only for use when birth date is not known.
disappeared_date	(For missing people) Date of disappearance: disappeared date and age (if birth date is known).
death_date	Date of death: death date and age (if birth date is known) or death date (if birth date unknown).
education	Education, e.g. degree, institution and graduation year, if relevant.
years_active	Date range in years during which the person was active in their principal occupation(s) and/or other activity for which they are notable.
era	Era in which the person lived and worked; less specific than years_active=. e.g. "Medieval"
term	Years the person held to a title, including <i>Formal title</i> , such as First Lady of Japan for Akie Abe, <i>Awarded title</i> , such as Mr. Olympia for Arnold Schwarzenegger, and <i>Job title</i> , such as President of Calvin College for Anthony Diekema.
spouse(s)	Name of spouse(s), followed by years of marriage, e.g. Name (m. 2014–present) for current spouse and Name (1970–99) for former spouse(s)
partner(s)	Name of unmarried life partner(s). Same format as spouse(s).

Table 3.2: Summary of infobox parameters for person entities in Wikipedia

3.3 Preprocessing of data

Before information can be extracted from the data, the data must be preprocessed to prepare it for further processing. There are several operations that can be applied for document preprocessing, including lexical analysis, elimination of stopwords, stemming, keyword selection and construction of thesauri. This section will elaborate on the subset of these techniques that are relevant for this thesis.

3.3.1 Lexical Analysis

Lexical analysis, or tokenization, is the process of segmenting a text into units known as **tokens**. According to definitions by Manning et. al [49], a *token* is an instance of a sequence of characters that are grouped together as a useful semantic unit, often referred to as a **syntactic word**, which includes systematically undoing contradictions, as in "don't = do not", and to define rules on how to handle multiword lexeme. A **type** is the class of all tokens that contains the same syntactic word, while a **term** is a normalized type.

(3.2) To be or not to be, that is the question.

In example 3.2 there are twelve tokens, including punctuation, but only ten types because there are two instances of each of the tokens *to* and *be*.

Token normalization is the canonicalization of tokens so that tokens matches despite of superficial differences. For instance, if you perform a web search for the string "USA", you might hope to also match documents containing "U.S.A". If both "USA" and "U.S.A" are mapped onto the same term, then a search for one of the tokens would retrieve documents that contain either of them.

(3.3) Alice and Bob doesn't like their Hewlett-Packard printer.

A major question in the tokenization phase is to decide what tokens are correct to use for distinguishing terms. Considering only example 3.2, tokenization may look trivial: Terms are split by whitespaces, and punctuation characters are removed. This would be a naïve approach of several reasons. Firstly, whitespaces may mean different things in different languages. For example, in languages like English and Norwegian, whitespace-tokens are often used to distinguish separate words. In contrast, other languages such as Chinese and Japanese does not use whitespaces at all, while in Hebrew and Arabic whitespace-tokens and their mapping to syntac-

tic words plays a central role and require a system that considers both tokens and syntactic words. This indicates that it may be expedient to use NLP tools that are intended for one particular language, and that the data being processed should be in the same language. Secondly, it does not account for the number of special cases in the English language. For instance, considering how to handle apostrophes, what is the correct tokenization of *doesn't* in example 3.3? Is it

- (a) *doesn't*
- (b) *doesnt*
- (c) *does n't*, or
- (d) *doesn t?*

If we assume that our initial proposal is correct, whitespace always indicates separate tokens. In reality some space-separated tokens are often viewed as single tokens. Examples include proper names such as *San Francisco*, foreign expressions such as *au fait* and compounds that are sometimes written as a single word and sometimes space-separated, e.g. *window sill* vs. *window sill*.

Another example is hyphenation, which is used for various purposes in English, including

- for splitting up vowels in words (*co-occurrence*)
- to show word grouping (the *hold-him-back-and-drag-him-away maneuver*), and
- for joining nouns as names (*Hewlett-Packard*)

Should these examples be interpreted as one token, or separated into words? Although most these examples are simple, they demonstrate why the initial assumption would not hold in a general context. Moreover, the issues are largely language-specific, which underpins our argument in only dealing with English documents. Also taking into account domain-specific properties and constraints such as special characters in programming languages or aircraft-names (B-52), tokenization becomes an even more important tool when it comes to entity identification. A practical example that illustrates this importance is given in figure 3.6 on page 50. Although tokenization is considered a relatively simple task in comparison with other tasks in NLP, but is nevertheless an important one, as any errors made during this phase will cause a ripple effect in all subsequent phases.

3.3.2 Eliminating Stopwords

If some words appear too frequent in a text or document collection, they are often deemed useless as discriminators for indexing purposes in information retrieval [50]. Such words are often referred to as **stopwords**, and includes common words in a language, such as grammatical articles, prepositions, and conjunctions. Eliminating stopwords has the benefit of considerably reducing the size of the index structure, and is therefore often common practice in search and ranking algorithms. However, although stopword removal may benefit from a reduced index size, it may suffer from reduced recall. Moreover, there is no single universal list of stopwords. This is because what may be considered a common word in one domain some may have a special meaning in another. The list of stopwords is therefore closely related to the content of the document or document corpus. Not all NLP tools have the need to use such a list, and some tools even specifically avoids removing stopwords, e.g. to support phrase search.

(3.4) Barlow predicts Take That reunion tour will feature only four members.

If we were to remove the stopwords from the sentence in example 3.3¹², it would return "Alice", "Bob", "Hewlett-Packard", and "printer". That is a good match for nouns, and could be a useful discriminator when searching for named entities. However, if we were to remove the stopwords from the sentence in example 3.4, we would remove important information because the name of the band "Take That" consists solely of stopwords, and thus is eliminated. As elimination of stopwords could remove useful information for our purposes, especially in the context of part-of-speech tagging, and phrase structure and dependency parsing, we therefore argue that it is futile, and potentially harmful for system accuracy, to perform stopword elimination, and will thus not apply this operation on our system.

3.3.3 Stemming and Lemmatization

For grammatical reasons, documents are likely to use different forms of a word, e.g. "organize", "organizes", and "organizing". Additionally, there are families of derivationally related words with similar meanings, such as "democracy", "democratic", and "democratization". In many situations, it could be useful if a search for one of these words returned documents that contained derivationally or inflectionally related words. Stemming and lemmatization are techniques that may be used to achieve this. **Stemming** usually refers to a process that substitutes inflected or derived words

¹²According to Wikimedia's online list of Google stopwords [51]

with their respective **word stem**, which is the portion of a word that is left after removing the word's prefixes and suffixes [50, 49]. **Lemmatization** differs from stemming in that its removal process often involves a vocabulary and morphological analysis of words, and that it returns the base form of a word, known as the **lemma**, rather than the word stem [49]. Additionally, lemmatization is able to match the lemma with a word with equivalent meaning, which usually is handled by a separate system in traditional stemming. An excellent example to illustrate the difference between stemming and lemmatization is how they cope with the word "saw": Stemming might return just "s", whereas lemmatization would attempt to return either "see" or "saw" depending on whether the use of the word was as a verb or a noun.

Both stemming and lemmatization aims to reduce extended forms of a word to a common concept. This could be useful for improving performance in information retrieval, but there are no general consensus about whether lemmatization and stemming actually leads to improvement in performance in information retrieval systems [52, 50, 53, 13]. One possible explanation on why performance could be reduced when stemming and lemmatization is applied, is that grouping the various forms of a stem or lemma not always yields good results, and in fact could lead to loss of information. This especially applies to collocations and cases where resulting word stem may not be related to the actual word. For example, it is more likely that a search for inflected variants of "operating system" would perform better than a search on all paragraphs containing "operat-" and "system".

- (3.5) Bing, Microsoft's replacement for Live Search, was unveiled by Microsoft CEO Steve Ballmer.

However, lemmatization is often applied in parsing contexts, as witnessed by annotator dependencies in appendix B. This is because proper names typically are stored in normalized forms e.g. for indexing purposes. To illustrate how this would affect performance of a named entity recognizer, consider the sentence in example 3.5. If lemmatization had not been applied, entity mentions on a genitive such as "Microsoft's" would not have been recognized as a named entity, in this case Microsoft, by an named entity recognizer.

3.4 Parsing

A general notion in machine learning is that the classification accuracy can be boosted if multiple classifiers are combined [49]. This is because the parsers often differ in their approach and underlying algorithms, which consequently leads to dif-

ferent parsing results. The idea is that whenever a sentence is parsed incorrectly by one parser, the same sentence might have been correctly parsed by another parser. Combining the results from three different parsers may reduce the probability of obtaining parsing-related errors, and thus approaches a collective disambiguation that applies across the system.

The parsers used in this project includes **SERIF**, which the corpus is tagged with, in addition to separate runs of the **MaltParser** and **Stanford CoreNLP parser**. The results from SERIF and MaltParser would then be converted to the same output format as the Stanford CoreNLP parser, and the union of their results would be used as input in DeepDive.

3.4.1 Stanford CoreNLP

Stanford CoreNLP¹³ is an integrated NLP framework that includes a set of core tools for natural language analysis developed at Stanford University. It follows a pipeline architecture that includes annotators such as a tokenizer, a part-of-speech-tagger, named entity recognizer, a parser and a coreference resolution system, which easily can be enabled or disabled by simply choosing to include or exclude them in the Stanford CoreNLP pipeline. Further details on the applied annotators may be found in appendix B. The Stanford CoreNLP tokenizer is based on the Penn Treebank 3 (PTB) standard¹⁴.

3.4.2 SERIF

Statistical Entity & Relation Information Finding (SERIF) is an extraction engine developed by BBN Technologies. It is the official document set for TREC KBA 2014¹⁵. According to [54], SERIF follow the PTB standard.

3.4.3 MaltParser

The **MaltParser**¹⁶ is a system for data-driven dependency parsing developed at Växjö University and Uppsala University in Sweden. Stanford used a English pre-trained model of MaltParser¹⁷ based on a linear Support Vector Machine (SVM),

¹³stanfordnlp.github.io/CoreNLP

¹⁴nlp.stanford.edu/software/tokenizer.shtml

¹⁵s3.amazonaws.com/aws-publicdatasets/trec/kba/index.html

¹⁶maltparser.org

¹⁷C. Zhang, E-mail correspondence (13th March, 2016). See maltparser.org/mco/mco.html

and trained on the Penn Treebank¹⁸. According to its website¹⁹, this configuration of the parser assume that its input is in the CoNLL format and is tagged with the Penn Treebank notation tagset, and outputs Stanford typed dependencies. As the Malt-Parser depends on input in CoNLL format, it is a prerequisite that the MaltParser in any case is run *after* the CoreNLP parser.

3.5 The DeepDive Framework

DeepDive is an Information Retrieval (IR) framework developed at Stanford University. The framework can be applied to process unstructured data, such as text documents, and from it produce structured data, such as SQL tables, which then is integrated with existing structured databases. On a more conceptual level, this entails extracting complex relationships between entities in text and making inferences about facts involving those entities.

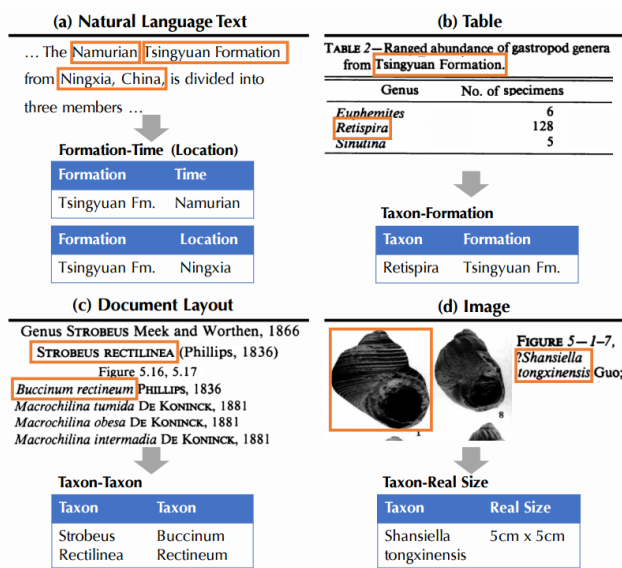


Figure 3.2: Illustration of the input and output of a Knowledge Base Construction system built for paleontology. Reprinted from Zhang[4]

¹⁸C. Zhang, E-mail correspondence (5th April, 2016)

¹⁹maltparser.org/mco/english_parser/engmalt.html

Figure 3.2 illustrates how DeepDive is used to construct a Knowledge Base (KB) to extract information from various sources of input.

3.5.1 DeepDive operations

This section will elaborate on four common DeepDive operations, adapted from [5] and illustrated in figure 3.3. The three latter operations may be run separately or combined, and can be applied for system calibration and for system results improvement.

Input

The first step is to prepare DeepDive for data by initializing the database and creating table schemas to extract the desired values from the data. DeepDive supports files with `.tsv`, `.csv` or `sql` extension, as well as serialized and compressed versions of these files, including files embedded in shell scripts.

Feature Extraction

When the data is loaded, DeepDive lets the users perform feature extraction by creating their own **UDF (user defined function)**, using any scripting language. Figure 3.3 (b) shows one example that uses Python to extract the word sequence between two entity mentions in a sentence. Assuming the first step is completed, the user is able to select desired input by using a SQL query, which in this case selects all sentences available in the uploaded data. The user's UDF is then executed for each tuple in the result from the preceding SQL query.

Constraints and Domain Knowledge

DeepDive allows the user to integrate constraints and domain knowledge as correlations among variables, illustrated in figure 3.3 (c). Imagine for example that a user wants to integrate a simple rule stating that a person is likely to be the spouse of maximum one other person. Given the entity "Barack Obama", the rule entails that only one of the relation candidates (Barack Obama, Michelle Obama) and (Barack Obama, Michelle Williams) could be true. The SQL query in figure 3.3 (c) obtains two relation candidates, each taking two variables. The first variable is a primary entity, which is similar in both relation candidates, while the second variable is a secondary entity that differs in the relation candidates. The function `AND(t0, t1)` defines how the variables correlate, whereas the weight defines how strong, or how

likely, this correlation is. In this case, the rule indicates that it is unlikely that both of the relation candidates are true. This can be executed with other rules, and more or fewer variables, to generate weight scores, which DeepDive can apply to learn probabilities from the data.

Distant Supervision

DeepDive follows the standards of the distant supervision assumption, which is that if a sentence contains two entities, the sentence may express a relation between the entities. This makes it easier to incorporate data, but it could also potentially create a lot of noisy data because it assumes relationships that may not exist.

We attempted to apply distant supervision in DeepDive by downloading a DBpedia dataset containing Wikipedia disambiguation pages and its associated entity pages, and using this to make a selection of ambiguous entities from the document corpus. This approach did however turn out to be unprofitable for our purposes because it matched next to none entities in the document corpus, and was eventually scrapped. We do however believe that the approach may bear fruits in the event that the entire document corpus was processed, as it may result in a relevant dataset that may be used for testing purposes.

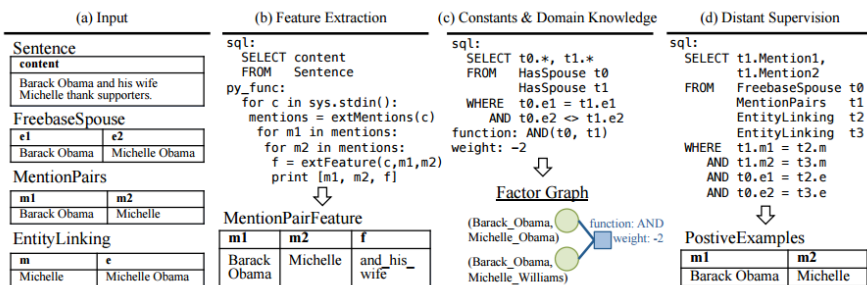


Figure 3.3: DeepDive operations. (a) Prepare datasets in relational form. (b) Generate labels using distant supervision with SQL; (c) Integrate constraints with SQL and logic functions; (d) Extract features with SQL and script languages. Reprinted from [5]

3.6 Challenges

This section describes the challenges that was encountered during the processing of the data set and development of the system.

3.6.1 Nouns Not Considered as Potential Named Entities

It is not unreasonable to think that most humans would be able to reason that the sentence in example 3.6 is about a particular named entity, and not just about any president. However, the Stanford NER does not recognize "president" as an entity because it is just a regular noun, and not a proper noun.

- (3.6) The U.S. president visited Norway in 2009 when he received the Nobel Peace Prize.

The U.S. president visited Norway in 2009 when he received the Nobel Peace Prize.

Potential tags:

LOCATION

ORGANIZATION

DATE

MONEY

PERSON

PERCENT

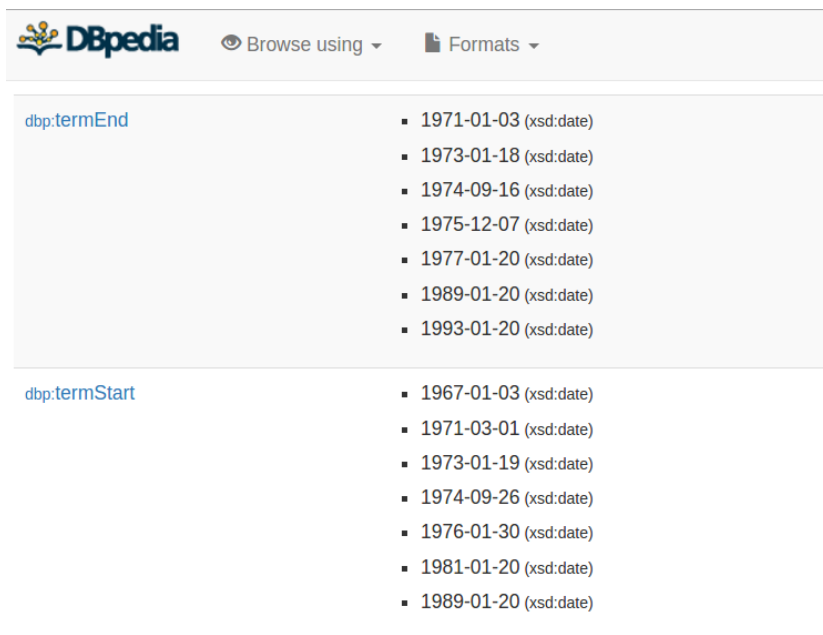
TIME

Figure 3.4: Screenshot from sample sentence parsed with Stanford NER

Figure 3.4 demonstrates that only LOCATION, ORGANIZATION and DATE are recognized as potential named entity categories by the Stanford NER. However, though not shown in the visual representation of Stanford NER in the screenshot above, the words in the sentence that has *not* been tagged by the named entity categories listed in the figure are actually tagged by another named entity category, simply referred to as *O*, or *OTHER*. Although we could include the OTHER category, that could entail also including typical stopwords (see section 3.6.1), which would involve part-of-speech analysis to identify nouns that could be affiliated with proper nouns, going well beyond the scope of this thesis. This aspect could however be interesting to explore in future work.

3.6.2 Multiple Values for Temporal Attributes

DBpedia was selected as knowledge base because it was the knowledge base alternative with the most datatypes, and was therefore thought to also provide the most temporal data types. However, closer investigations revealed that the only temporal data available in a superficial search of an person entity page on DBpedia was flat temporal attributes, i.e. attributes only corresponding to one value, such as the date of birth, and if the person was deceased, the date of death, ref. table 3.2 on page 36. This is however only given that the person is a real-world individual, and not a fictional character, which is not a distinction made by the Stanford NER nor our system.



The screenshot shows the DBpedia interface for George H. W. Bush. It displays two lists of dates, each with a corresponding property name and a list of dates in ISO 8601 format (xsd:date).

Property	Value
dbp:termEnd	1971-01-03 (xsd:date)
	1973-01-18 (xsd:date)
	1974-09-16 (xsd:date)
	1975-12-07 (xsd:date)
	1977-01-20 (xsd:date)
	1993-01-20 (xsd:date)
dbp:termStart	1967-01-03 (xsd:date)
	1971-03-01 (xsd:date)
	1973-01-19 (xsd:date)
	1974-09-26 (xsd:date)
	1976-01-30 (xsd:date)
	1981-01-20 (xsd:date)
	1989-01-20 (xsd:date)

Figure 3.5: Screenshot of term intervals for DBpedia entry George H. W. Bush. The list is sorted by date, and the term interval is given by a combination of the list elements in the `dbp:termStart` and the list item in `dbp:termEnd` at a corresponding position.

At this point, the available temporal data for a real-world person entity is 1) the stream time of the document, which we assume to be the time when the document was generated, 2) the entity's date of birth and possibly 3) the entity's date of death.

This information could really only convey whether or not the person was alive during the time the document first was generated.

(3.7) This week, president Bush announced that . . .

Returning to our presidential example from page 19, here repeated as example 3.7, this temporal information could not really clarify whether the sentence was about George W. Bush or George H. W. Bush, unless the stream time of the document was before George W. Bush was born. However, although the date of birth is the only temporal data we can usually assume to find about a person entity, there are other temporal data available for special cases, cf. table 3.2. For example, if an entity is currently holding, or has held, a political position such as president, mayor or governor, the date of when their term started and ended are usually provided. It is however important to note that these temporal data are likely to not just contain one value, but rather a list of values, as depicted in figure 3.5.

3.6.3 Unavailable Temporal Data

As we addressed in section 3.2, there may be cases where Wikipedia infoboxes are incomplete, and therefore lack temporal information that one normally can assume is available about person entities. For such cases, we suggest using cosine similarity to compare DBpedia entity pages for potential target entities with the document of which the entity mention resides.

3.6.4 Dataset Incompatibility

One of the first tasks in the preprocessing was to obtain a sample of the dataset that was to be used in the experiments, and use this sample to test whether decryption, decompression and deserialization was performed correctly. This sample came from the TREC KBA 2012 dataset. The deserialization was performed with a designated framework that specialized processing massive data streams, namely `streamcorpus.org`, as suggested by TREC.

However, deserialization proved to be a difficult task. The framework could not process the data. Further investigation revealed that the data did not match the input expected by the framework, i.e. variables designated for retrieving specific values, such as `clean_visible`, did not exist. This was caused by a technological shift in the development of the framework and technology used by the framework, most

notably the serialization technology Thrift. This entailed that the dataset was serialized with now-deprecated and thus incompatible technology, preventing the data to be correctly deserialized with the updated technology. These issues could most likely have been resolved by rolling the technologies back to previous versions and making alterations in the document content, but this would also have required a lot of resources in terms of time and computational power. Eventually, it was decided to renounce the TREC 2012 dataset and instead obtain a newer version of the dataset from 2014.

3.6.5 Errors Caused by Special Characters

Although `streamcorpus.org` claims that the `clean_visible` version of the dataset is correctly encoded with UTF-8, observations has been made of several instances of characters referring to the Unicode code point rather than the UTF-8 encoded character. It was also observed that because HTML tags were replaced with whitespaces, some of the documents consisted almost exclusively of whitespace characters, which was directly linked to the size of the documents, and moreover disturbed the parsing because some whitespace characters such as `\n` was interpreted as a part of the words, e.g. "`\nThese`". These issues was dealt with by encoding the content of `StreamItems` with ASCII, causing UTF-8-specific characters to be ignored and removed, and by replacing all cases with one or more preceding whitespaces with a single space.

During the entry of data in the PostgreSQL database, three types of error occurred: 1) End-of-line marker corrupt, 2) Literal carriage return found in data, and 3) Extra data found after last expected column. The first kind of error was caused by occurrences of `"\"` (backslash) in the text that was not escaped with an additional backslash. Additionally, the combination of a backslash and a period in the text is a way to represent end of data in the PostgreSQL database²⁰. The second kind of error was caused by using the backslash character when the slash characters should have been applied, causing the backslash character in the text to be transformed into a literal carriage return, represented as `^M`. The third error was caused by escape characters in the text that made the database assume that the row was completely processed. All of these errors were handled by deleting the line in question, but in order to keep as much data as possible in future work, we suggest defusing the characters causing the errors rather than deleting the line where the error resides.

²⁰Documentation available at postgresql.org/docs/9.2/static/sql-copy.html

3.6.6 Unifying Parser Results

The first phase of the union was to select a small sample from the different document categories, and then compare the results from each parser to clarify what adaptations had to be made to make the output compatible with DeepDive. This phase included:

1. Retrieving the `clean_visible` from the TREC dataset and parsing it with Stanford CoreNLP
2. Use the output from Stanford CoreNLP as input in the MaltParser
3. Retrieving relevant columns from SERIF

As named entity categories are an important focus of this thesis, the first comparison concentrated on the word form and named entity category, known as `FORM` and `NERTAG` in technical terms. The comparison led to two imperative discoveries:

1. The results from the MaltParser and Stanford CoreNLP parser were identical.
2. SERIF and Stanford CoreNLP use different standards in tokenization.

These findings forced us to reconsider whether it was still expedient to use three parsers. The first finding indicated that it may be unnecessary to run both the Stanford CoreNLP parser and MaltParser, at least in the case of the `FORM` and `NERTAG` columns, as these columns were identical to that of the other parser. One obvious reason to drop the MaltParser in favor of Stanford CoreNLP is that the MaltParser input depends on output from Stanford CoreNLP.

With this in mind, we assume that the aforementioned reason excludes MaltParser as a viable option, and move on to the second finding. Although both Stanford CoreNLP and SERIF allege that they follow the Penn Treebank (PTB) notation for the tokenization phase, indisputable differences in token interpretation between Stanford CoreNLP and SERIF was observed. According to the official Penn Treebank website²¹, there are rules that outlines how text is tokenized. Highlights includes verb contractions and special characters such as punctuation, hyphens, and various types of parentheses. The tokenizer in Stanford CoreNLP, implemented in

²¹[HTTPS://www.cis.upenn.edu/treebank/tokenization.html](https://www.cis.upenn.edu/treebank/tokenization.html)

PTBTokenizer²², follow PTB notation according to these rules. The technical details described in the PTBTokenizer documentation emphasize that although SERIF claim to use the PTB notation for tokenization [55], it diverges notably from the PTB standard in at least two cases.

20	(CNRS	ORG	20	-LRB-	O
21	UMR8022	None	21	CNRS	ORG
22)	PER	22	UMR8022	O
23	slawomir.staworko	PER	23	-RRB-	O
24	@	None	24	slawomir.staworko@inria.fr	O
25	inria.	LOC			
26	fr	None			

(a) SERIF

(b) Stanford CoreNLP

Figure 3.6: Excerpt that has been parsed by SERIF and Stanford CoreNLP

For example, the PTB standard is to map bracket-like characters to special three-letter sequences, e.g. the round parentheses '(' and ')' to -LRB- and -RRB-, respectively. As illustrated in figure 3.6, this standard has been followed by Stanford CoreNLP, but not by SERIF. Moreover, note that Stanford CoreNLP on line 24 in figure 3.6 (b) identifies the e-mail address as a single token and assigns the NER-tag O (OTHER). In SERIF the same character sequence is split into four tokens and then assigned the NER-tags PER (PERSON), None, LOC (LOCATION) and None. None of the parsers produce a optimal result in this instance, which would be to treat `slawomir.staworko@inria.fr` as a single token and NER-tag it as an e-mail, but Stanford CoreNLP certainly came the closest to the desired outcome. On a further note, SERIF appears to be both too eager to assign entity categories to tokens that are quite ordinary English expressions (for example, it interprets *i.e.* to be a location), at the same time as it does not recognize numerical or temporal expressions.

As merging the results from these parsers requires more resources than first anticipated, it becomes a question of not only practicability within the available time scope, but also whether it is expedient to use more than one parser, especially if one of the parsers outperforms the other. On the account that named entity categories

²²Available at nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/process/PTBTokenizer.html

are an important focus of this thesis, it is therefore pertinent to select the parser that surpasses the other parser in NER-tag accuracy and selection, with a special emphasis of the parser's ability to recognize temporal expressions. Based on the submitted evidence and details of the Stanford NER in section 2.5.3, Stanford CoreNLP clearly stands out as the better alternative and was thus selected to be the parser applied in the experiments.

However, we do recognize that information may be gained from document context and surrounding text, represented in data columns such as POS-tags and dependency relations, and will therefore discuss briefly how these data columns combined with multiple parsers could be exploited in future work in section 6.2.

In this chapter, we describe how the experiments were set up, and how they were evaluated. Lastly, we represent the experiment results.

4.1 Experiment Setup

This section describe the dataset that used in the experiments, and the methods that was used to evaluate the experiments.

4.1.1 Experiment Outline

The experiment can be divided into these following steps:

1. Preprocess dataset
 - (a) Select sample of dataset
 - (b) Decrypt, decompress and deserialize the sample
2. Parse preprocessed dataset
3. Extract entities from parsed dataset with information extraction framework
4. Link extracted entities in the dataset to an external knowledge base
5. Evaluate experiment results

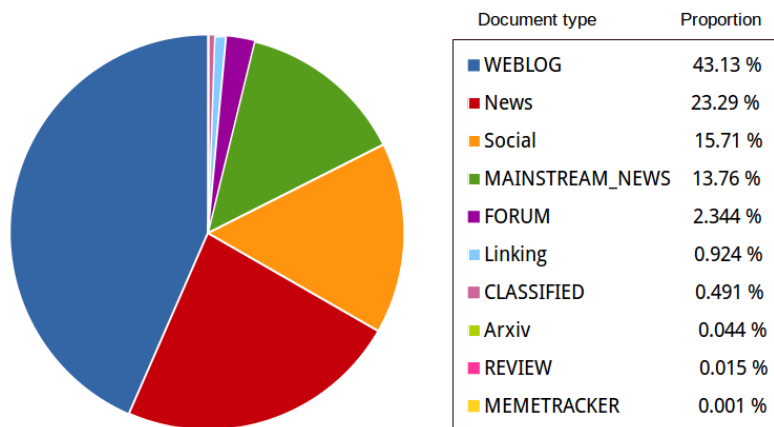


Figure 4.1: Proportion of data sources from the TREC 2014 Stream corpus

4.1.2 Evaluation Methodology

Dataset

The experiment use the stream corpus from the Text REtrieval Conference (TREC) Knowledge Base Acceleration (KBA) 2014 [56], which is a representative and well-established document collection, extensively used in prior work of name entity disambiguation. The official TREC dataset consists of approximately 580 million documents semantically tagged by the SERIF system [57]. The total size of the dataset after compression and encryption amounts to approximately 10,9 TB. The dataset is divided into ten different document types and span in publication time from December 2011 to April 2013. The distribution of document types is depicted in figure 4.1.

Stream Corpus Structure

Figure 4.2 provide a conceptual understanding of how the stream corpus is organized: (1) The core is a *StreamItem*, representing a snapshot of a single document at a single point in time, and uniquely identified by its URL. (2) Several *StreamItems* are serialized into hourly chunks with Apache Thrift. A chunk range in size from a few hundred to a few hundred thousand *StreamItems*. The resulting chunks are compressed with XZ (3) and then encrypted with GnuPG (4).

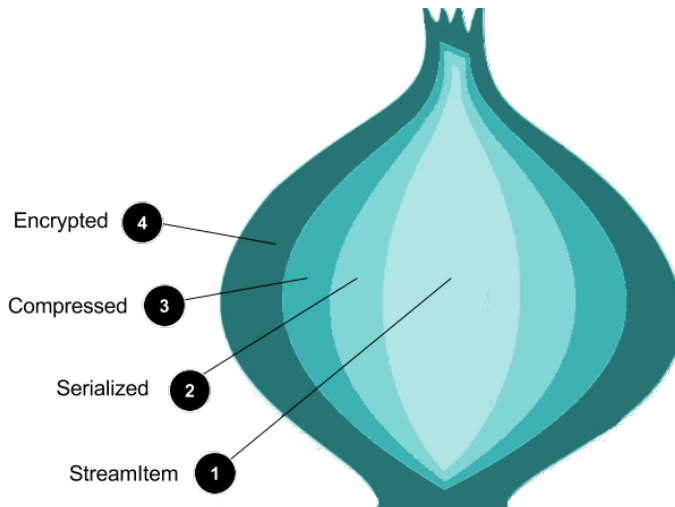


Figure 4.2: The Stream Corpus Onion

The StreamItems are deserialized with the StreamCorpus python-package¹. Figure 4.3 depicts a graphic model of how StreamItems are organized. Each StreamItem contains metadata such as `stream_id`, `doc_id`, `stream_time`, and `body`. Inside `body`, there is a `ContentItem`, which is a complete representation of the StreamItem’s data. It includes several properties that describes the data, as well as different versions of the data, which is further elaborated in table 4.1.

Version	Description
<code>raw</code>	the original file, downloaded as an unprocessed byte array
<code>clean_html</code>	HTML-formatted version of <code>raw</code> , with correct UTF-8 encoding and no broken tags. HTML-escaped characters are converted to their UTF-8 equivalents. <code><</code> , <code>></code> , and <code>&</code> are escaped.
<code>clean_visible</code>	copy of <code>clean_html</code> where all HTML tags is replaced with whitespace. Can be directly inserted into an HTML or XML document without further escaping.

Table 4.1: Summary of data versions

¹Documentation available at streamcorpus.org

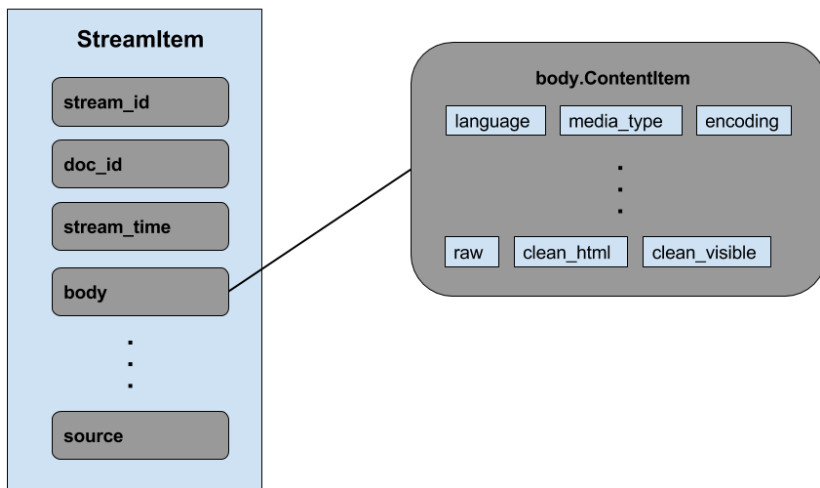


Figure 4.3: Structure of a StreamItem

Constraining the Dataset

Due to language-specific properties, only the subset of English documents in the corpus is relevant for this thesis. Fortunately, TREC provides a subset with all non-English documents removed and `StreamItem.body.raw` set to an empty string, which reduces the dataset to just over 500 million StreamItems distributed over 11.948 chunks, and 4,5 TB in size. To help distinguish between the two datasets, the aforementioned subset is referred to as dataset *B*, while the original dataset is referred to as dataset *A*. Although *A* is reduced to less than half its original size, dataset *B* is still of such a size that it introduces challenges when it comes to time and computational resources.

As this system is a proof of concept, we argue that it is sufficient to process a subset of dataset *B* to prove its feasibility. The first step to attain this subset, dataset *C*, was to generate a list of the most recent chunks. As there was no way to know in advance how the StreamItems would be distributed and consequently nor the total size of the data, 10 percent of the chunks from subset *B* were retrieved, based on their date novelty. The oldest chunks were then removed from our generated list to adjust for our available resources. This new dataset *C* amounts to 305.987

StreamItems distributed over 1190 chunks, and approximately 492 GB when the data is encrypted, compressed and serialized.

After the data was retrieved and the number of chunks was reduced, the data was prepared for parsing through decryption, decompression and deserialization of the files. As this preprocessing progressed, intermediate files were deleted to further reduce the size of the data. Preprocessing and parsing was tested on a substantially smaller dataset for performance optimization, and yielded promising results. When applied to the actual dataset C , however, it turned out that the system was able to process on average just over 200 StreamItems per day. Should we have processed the full dataset C with our available resources, the preprocessing and parsing alone would have taken over four years to complete. Due to constraints in time and resources, we therefore selected a subset of dataset C , which was the resulting dataset after running annotation for five days. We will refer to this subset as dataset D . It amounts to 1,7 GB when encrypted, compressed and serialized, 6,7 GB postprocessed, and approximately 5 million sentences, each representing an entity mention.

Note that only a subset of the content of each StreamItem were retrieved, namely `body.clean_visible` and `stream_id`. These fields are required because the `body.clean_visible` is the HTML-escaped raw document text to be parsed by Stanford CoreNLP, and the `stream_id` serves as a unique identifier for the StreamItem.

It is usually possible to parse the documents in DeepDive. However, due to technical difficulties, this step had to be performed separately, using the same parser as the one embedded in DeepDive. Although it was not ideal, it did provide us with the opportunity to configure the parser for improved performance, we did for example not need to run *all* the default annotators. See appendix B for further details.

When the preprocessing and parsing was complete, the parsed data was loaded into DeepDive and stored as a PostgreSQL database table. Each tuple in this database table is a sentence, uniquely identified by its `stream_id` and `sentence_index`. Additionally, each sentence contains the sentence's text, as well as its tokens, lemmas, POS-tags and NER-tags.

4.1.3 Baseline Methods

When the data has been preprocessed, parsed and loaded into DeepDive, the next step is to extract entities that has been recognized as entities by the Stanford NER. In principle, we could choose all tokens that has been tagged with one particular named entity category, or that has been tagged with any named entity category².

In our experiments we have chosen to only extract `PERSON` entities. This is because the number of `PERSON` entities clearly surpasses that of other entity types in DBpedia, in addition to the fact that `PERSON` entities are the most likely to be associated with great amounts of temporal data compared to other entity types because it is in the `PERSON` entities' nature to change over the course of their lifespan.

In DeepDive, we extract mentions of `PERSON` entities by first creating a database schema, `person_mention`, containing the mention, stream identifier, sentence index and sentence text, and then creating a function that returns tuples conforming to the `person_mention` schema. This function calls upon a User Defined Function (UDF), which finds phrases that are continuous tokens tagged as `PERSONs`. Then, we specify a new function that is applied to the tuples in the `sentences` database table and applied to the `person_mention` table.

The next step is to establish a link between the `PERSON` mentions in the documents and a knowledge base.

```
(4.1)  mention  stream_id  index  text
        Sansa      1351c...7aa411  13      But, at the end, Sansa was the danger ...
```

First, we copy the `person_mention` table from DeepDive's underlying database onto a file. In this file, each line contains data as structured in example 4.1. The database contains 2.009.101 entity mentions, distributed over 1.051.648 unique sentences. A naïve way to establish links between all the entity mentions and their corresponding entity page in DBpedia is to perform a lookup in DBpedia for each of the entity mentions in the input text. However, because we do not want to do unnecessary lookups, and in worst case scenario even cause a Distributed Denial of Service (DDoS) attack and get our IP address blocked from DBpedia, it is expedient to reduce the number of lookups. We reduce the number of lookups by adding all entity mentions to a separate list, and then removing entity mention duplicates,

²Excluding the `OTHER` category, which is used for tokens that does not qualify as entities

shrinking the number of lookups from 2.009.101 to 294.403.

To further optimize the lookup process in the knowledge base, the processing is divided into three steps: 1) preprocessing of all the entity mentions, 2) lookup of all preprocessed entity mentions, and 3) processing of the results from the lookup. Figure 4.4 shows the flow of this parallel entity linking. The reasoning behind this division of labour is that if entity mentions are processed separately, it increases the cost because the system has to wait for the HTTP request to respond before it can start on the next request, for every entity mention.

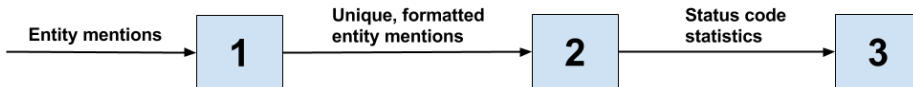


Figure 4.4: System flow in parallel entity linking

However, as the HTTP requests for each entity mention does not depend on each other to produce a response, they can be done in parallel, and thereby improve the system performance. Parallelizing the requests reduced the experiment runtime from approximately four days to two hours.

To check whether DBpedia has a page about a entity mention, we combine a URL-prefix, `dbpedia.org/page`³, with the entity mention, and assign this combination to a new variable, `entity_url`. A HTTP request is then sent with `entity_url` as input parameter to check whether the request returns a successful HTTP response code. This gives a complexity of $\mathcal{O}(m^2)$, which is the number of the number of StreamItems times the number of entities.

To avoid overloading the DBpedia server by bombarding it with HTTP requests, the list of unique entity mentions is divided into chunks of 300 entity mentions. HTTP requests is then sent in parallel to DBpedia for every entity mention in the chunk. When the processing of a chunk is complete, the system begin processing the next chunk. This way, no more than 300 HTTP requests are processed simultaneously, preventing our own system from crashing, as well as avoiding launching an inadvertent DDoS attacks.

The HTTP request may return one of six responses, listed in table 4.2. If any status

³`dbpedia.org/page` and `dbpedia.org/resource` are equivalents

code other than 200 is returned, we assume the request was unsuccessful and that the entity mention in its current form is not in DBpedia. We will use the number of successful HTTP responses to determine the accuracy of our entity linking system.

Status code	Name	Description
000	Unknown Error	An error caused by connection drop and associated timeout
200	OK	Standard response for successful HTTP requests
301	Moved Permanently	Response for permanent URL redirection
404	Not Found	The requested resource could not be found
500	Internal Server Error	Generic error message, given when an unexpected condition was encountered and no more specific message is suitable
502	Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server
503	Service Unavailable	The server is currently unavailable (because it is overloaded or down for maintenance)

Table 4.2: Description of status code responses returned by HTTP request

4.1.4 Evaluation Metrics

System Accuracy

To measure the accuracy of the system in a classification context, we use the standard precision, recall and F-measure based on the values given in the confusion matrix in table 4.3.

		Predicted class	
		+	-
Actual class	+	True positive	False positive
	-	False negative	True negative

Table 4.3: Confusion matrix for a binary classification problem

Here, the "Predicted class" refer to a classifier's predicted results, while "Actual class" refers to the actual classifier results. True and false refers to whether the prediction corresponds to the observations. In our case, we have a set of sentences that contains at least one `PERSON` entity tag. We measure the accuracy of the entity linking by checking if the `PERSON` entity tag combined with a URL-prefix return a successful HTTP response code. This can be interpreted as:

- **True Positive (TP):** Sentence was retrieved and returned a successful HTTP response code
- **False Positive(FP):** Sentence was not retrieved but returned a successful HTTP response code
- **True Negative (TN):** Sentence was not retrieved and did not return a successful HTTP response code
- **False Negative(FN):** Sentence was retrieved but did not return a successful HTTP response code

From these definitions, precision and recall are defined as:

$$(4.2) \quad Precision = \frac{TP}{TP + FP}$$

$$(4.3) \quad \text{Recall} = \frac{TP}{TP + FN}$$

The F-measure is a harmonic mean of precision and recall. It has a parameter that sets the trade-off between recall and precision. The standard F-measure F_1 assigns equal importance to recall and precision, and is defined as follows:

$$(4.4) \quad F_1 = \frac{2TP}{2TP + FP + FN}$$

Cosine Similarity

When an ambiguous entity occur in a document, the entity points to a set of links that represents its potential target entities. Cosine similarity evaluates the degree of similarity of a document d_j with regard to a query q as the correlation between the vectors \vec{d}_j and \vec{q} . This correlation can be quantified by the cosine of the angle between these two vectors. In our case, q is some document from our text corpus, and d_j is a link to a potential target entity on DBpedia. The resulting similarity ranges from 1 to 0, where 1 means exactly the same, and 0 indicating no similarity.

$$(4.5) \quad \text{sim}(d, q) = \frac{d_j \bullet q}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^t w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^t w_{i,j}^2} \times \sqrt{\sum_{i=1}^t w_{i,q}^2}}$$

4.2 Results

Twenty experiments were conducted on the same set of entity mentions. In each experiment, the entity mentions were grouped into chunks of 300 instances. HTTP requests were then sent in parallel per item in the chunk. This was done to rule out any discrepancies caused by e.g. server downtime or HTTP request timeout.

From our previous definitions, only HTTP requests with HTTP response code 200 are deemed successful requests. Precision scores were therefore calculated with the set of instances that returned successful HTTP response codes as True Positives, and the entire set of unique entity mentions as the sum of True Positives and False Positives. The distribution of HTTP response status codes and calculations of precision scores for each experiment is shown in appendix A, in table A.2 and A.1, respectively.

The experiments revealed clear deviations in the distribution of HTTP responses on the same dataset. Although it was possible, it was highly unlikely that there would be that many and rapid changes on the same set of person entities in DBpedia within the approximate two days it took to conduct the experiments. A more reasonable explanation for the fluctuations in the results was that the server did not handle the requests properly, which in turn affected the resulting response codes. However, there was a chance that the experiments returned successful HTTP responses for *different* entities in each experiment. To explore that possibility, the intersection of successful HTTP response codes were extracted from the experiment log files, i.e. if a entity mention returned HTTP response code 200 in any of the experiments, the counter increased. This resulted in overall 85.206 successful response codes of 294.403 entity mentions, and a precision score of 28,942 %. In contrast, the average precision score for all the experiments was 14,652 %, ranging from 12,267 % to 16,282 %.

In this chapter, we will discuss the experiment results in the light of our research questions.

5.1 Extracting All Entities

The natural language text from our dataset is produced by humans, and thus it is not uncommon for human errors to occur, such as misspellings, or accidental removal of letters or spaces. Nor is it unusual for people to refer to other people by nicknames, alternate names, abbreviations, and by using lowercase letters instead of a correctly capitalized full entity name. This adds an element of uncertainty of how many of the entity mentions that actually exists in the data versus how many of them that are actually retrieved, because we cannot be sure that the Stanford NER will be able to identify person entities represented in such a form.

To determine whether there are person entities that should have been retrieved but were not, i.e. the set of FPs, a manual check would have to been conducted by an independent domain expert, or the statistics could have been provided by the TREC organizers. However, manual processing of the dataset would have been resource demanding beyond our capacities, and even if we had access to dataset statistics, it would only be useful to us if either the full dataset was processed, or if the statistics were mapped to specific chunks, on the account that we only processed a subset of the full dataset. Hence, from the data that is available to us, we cannot be entirely certain that we have indeed found all the FPs, which makes it impossible to calculate recall, and consequently also the F_1 -score, because it relies on recall.

5.2 Accuracy of Stanford NER

When a web document is annotated, *all* the textual content of the document is processed. Apart from main content, e.g. article text, blog post or discussion forum thread, it also includes scripts, menus, advertisements and spam, which may produce noise in the data.

Moreover, named entities are often identified by the considering adjacent capitalized proper nouns. Having this as the sole criteria could however cause the Named Entity Recognizer in the Stanford CoreNLP parser to assign named entity categories to tokens and phrases that are not actual named entities. This could prove problematic if the resulting named entities are applied directly for entity linking due to DBpedia URL standards.

#	Reason	Example
1	Capitalization	"MATT_BELLAMY", "JOHN", "AARON", Moist_Von_Lipwig
2	Non-alphabetic characters	"-_Max-_Ehrmann", "Kim_Leadley/Matthew_Briggs", "Anil_ji_-LRB-_Kapoor", "-_Liliana_11:15", "ISBN_978-0-9765531-3-7_Schuller", "O'Cantler"
3	Name repeated	"Farah_Al-Qasimi_Farah_Al-Qasimi_Farah_Al-Qasimi"
4	List of people	"Obama_Barack_Obama_Abraham_Lincoln", "Kristen_Stewart_Cheryl_Cole_Kate_Beckinsale_Jessica_Alba_Katy_Perry_Michelle_Keegan_Miranda_Kerr_Victoria_Justice_Taylor_Swift"
5	Misinterpreted abbreviations	"Ricardo_Henson_confirmed._Nathan_Rombley", "Paul_-Lester-_guardian.-_co_uk"
6	Foreign words	"Kostenloser_Versand_Deutschlandd", "Eigenhandel_von_Banken_einen_Riegel"
7	Foreign names	"Bjorn_Kjos", "Bjrn_Borg"
8	Nicknames	"Dyana_Shoppaholiccccc", "Vijay_S", "Sassy_Shanon_Todd_Tracy"

Table 5.1: Examples

According to DBpedia documentation¹, each thing in DBpedia is denoted by a URL on the form `dbpedia.org/resource/Name`. The requirements for these URLs are rigorous, and thus any URLs that does not correspond to its expected format is rejected.

¹wiki.dbpedia.org/services-resources/datasets/data-set-39

The combination of DBpedia's strict URL policy and direct application of person entities identified by the less discriminating Stanford NER consequently produce an entity linking system with a low success rate. In general, DBpedia does not recognize entities that diverges from their URL policy unless the entity mention already exists as an alias or alternate name that redirects to the correct entity page. Table 5.1 present some possible explanations on why some HTTP requests were unsuccessful, using observed patterns in entity mentions as examples.

5.2.1 Capitalization and Non-Alphabetic Characters

As the Stanford NER does not place particular high demands with regard to identifying named entities, the resulting set of potential named entities could be quite extensive. It includes all capitalized entity mentions, and entity mentions where parts of the name is wrongly capitalized, as observed in #1, in addition to entity mentions with non-alphabetic characters that are misplaced or unlikely to be part of a name, such as a slash, a time or an ISBN-number, as observed in #2.

5.2.2 Lack of Length Restrictions

Because there are no restrictions on the length of a name, it may be difficult to decide whether a person entity is represented using multiple names, or if it in fact is a list of named entities or a set of links without delimiters between each item, that is interpreted as one single entity mention, as observed in #3 and #4. The behaviour may be caused by a lack of delimiters between consecutive person entity mentions, but nonetheless, it might suggest to revisit the UDF for person entity extraction in DeepDive, and consider stricter rules on what constitutes a person entity.

Moreover, abbreviations are not uncommon in entity names. For example, the 43. American president is typically referred to as George W. Bush. However, the Stanford NER has not set any restrictions on length of abbreviations, resulting in cases such as those shown in #5.

5.2.3 Foreign Words and Names

Although we have assumed that our dataset consist solely of documents in English, some occurrences in Spanish and German have been observed. This is unfortunate because syntactic and semantic rules in other languages often differ from English, and may result in undesirable results in NLP tools that specialize in English language-properties. For instance, German capitalize all nouns, which could produce

errors if a Named Entity Recognizer expect English text, and thus wrongly categorize phrases that may not contain proper nouns at all. For example, the Stanford NER identifies the phrase "Kostenloser_Versand_Deutschland" as a person entity, but it actually translates into something like "Free shipping Germany". Moreover, if a person entity has a name that contains special characters that is not part of the English alphabet, these letters may be misinterpreted or removed, e.g. "Bjørn_Kjos" become "Bjorn_Kjos", and "Björn_Borg" become "Bjrn_Borg".

5.3 Data sources

A diverse and representative dataset is often crucial in information extraction applications because the results heavily depend on the content of the data sources. Our system is no different in that respect. For example, we expect the documents we process to be UTF-8 encoded, cleansed for HTML tags, and to only be in English.

However, in entity-centric systems, it is important to keep in mind that not all types of web documents are likely to contain information about person entities. For example, it is probable to find information about academic concepts in scientific articles, rather than factual information about particular people, perhaps except references to authors that is relevant to the article. Similarly, people may be referred to by nicknames in discussion forums, and bloggers may write about family, friends and colleagues.

Although person entities may be identified in such web documents, it is not unlikely that these entities are ordinary people without dedicated DBpedia pages, interns or scientists yet to make their breakthrough, or anonymous people that hides behind nicknames.

5.4 Scalability

Dataset D amounts to 6,7 GB postprocessed, and corresponds to about 5 million entity mentions. Considering that it took five days to generate this modest amount, and that we have access to a dataset amounting to 500 million StreamItems in 4,5 TB of data, i.e. more than 2.700 times larger than the dataset we currently use, it is clear that it would be expedient to scale our system so that it would be able to accommodate larger amounts of data. However, for it to be feasible to annotate the corpus and run our experiments within reasonable amount of time, we would need

more than one computer.

Let us consider what we would need to process 4,5 TB of data: First, 500 million StreamItems would have to been annotated. As StreamItems vary in length, we would not have concrete numbers on the amount of person entity mentions, but assuming that the proportion of entities are similar to that of dataset D , it will amount to about 13,5 billion person entity mentions. Then, still assuming that the relationship between the number of person entity mentions and unique person entity mentions are of similar proportions as in dataset D , the number of unique person entities would amount to about 2 billion. For every unique person entity, there would be a number of links to potential target entities. If we then chose to exploit the temporal data available to us in the knowledge base, there would also be a list of temporal data attributes affiliated with each potential target entity, with each temporal data attribute a potential list. This approach would have a complexity of $s \times e \times l \times t \times t_l$, where s is the number of StreamItems, e the number of entities, l the number of links to potential target entities, t in the number of temporal data attributes, and t_l is the number of values in each temporal data attribute. In a worst case scenario, the complexity would be $\mathcal{O}(m^5)$. A suggestion on how this could be dealt with will be further addressed in section 6.2.

5.5 Entity Linking Quality

Comparing the results from our entity linking experiment with the results from TAGME, it is clear that they are based on different models and preconditions. For the sentence in example 5.1, TAGME returns links for "Sansa" and "Hound", as depicted in figure 5.1. Moreover, "Sansa" has been identified as the line of flash memory-based portable media players *Sansa*², produced by SanDisk, while "Hound"³ has been recognized as a type of dog. In contrast, our system returns links for "Sansa" and "Sandor", where "Sansa" links to a disambiguation page, and "Sandor" links to a page with a list of men by the name Sándor⁴.

(5.1) "It is a way of telling that Sansa should protect herself of The Hound and not of Sandor."

An ideal entity linking system would recognize "Sansa", "The Hound" and "Sandor" in example 5.1 as entities, and attempt to link them to the corresponding page in an

²en.wikipedia.org/wiki/SanDisk_Sansa

³en.wikipedia.org/wiki/Hound

⁴dbpedia.org/page/S%C3%A1ndor

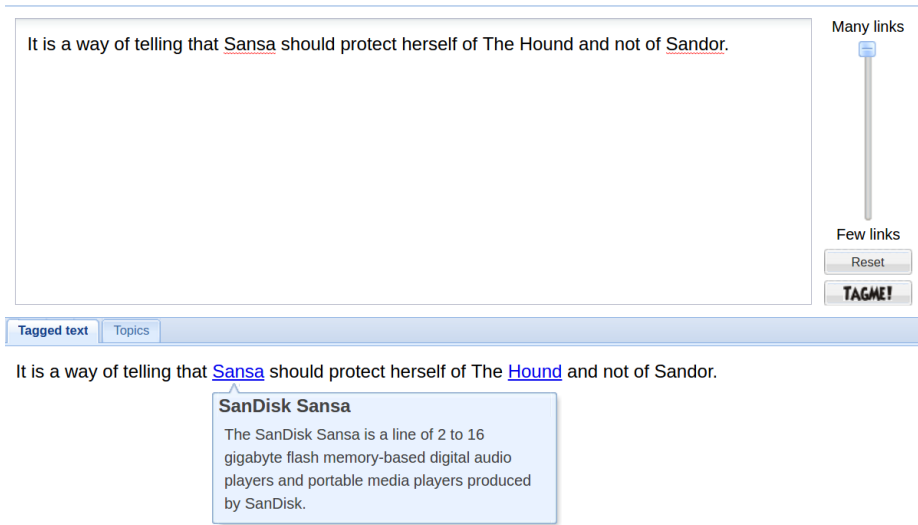


Figure 5.1: Screenshot of TAGME

But, at the end, Sansa was the danger to The Hound.

It is a way of telling that Sansa should protect herself of The Hound and not of Sandor.

I thought Andrew Garfield and Emma Stone were fantastic.

Which is weird because that will put LF and Varys on the same side....and it really depends on v first with the fresh army of the Vale.....same thing with her and Aegon, yet I feel with Aegon ba Ser Dylen of the Green Oak Freerider Members 31 posts Posted 01 November 2012 - 07:09 PM Lion Of Judah Lion Of Judah You Cannot Outrun The Pride.

The Vale DID have a side... - General (ASoIaF) - A Forum of Ice and Fire The Latest News Ste the Seven Kingdoms Lands of Ice and Fire: A Slice of Essos Three New GRRM Audiobooks W Peek at Braavos Vulture Dubs Most Devoted Fans Connect with Us Follow @westernsora Not

Figure 5.2: Screenshot of entity linking

encyclopedia or knowledge base such as Wikipedia or DBpedia. In cases where an entity were not linked to an actual article, such as a page containing a list of outgoing links to actual articles, the system could try to attach "`_ (disambiguation)`"⁵, to the the entity mention, and then again attempt to link it to the corresponding page in a knowledge base.

5.6 Limitations Imposed by System Design

Our approach assume that if a HTTP request for an entity mention returns a successful HTTP status code, the entity mention was linked to its corresponding entity. However, although this would mean that the entity mention in its current form is represented in the knowledge base, it is uncertain whether the URL actually refers to a disambiguation page, an entity page, or even whether the entity mention is a person entity.

(5.2) *Paris* and *Nicole* appeared in *The Simple Life*

If our system was to process the sentence in example 5.2, it would have recognized "Paris" and "Nicole" as person entities, but during the entity linking process, it would return a disambiguation page for "Nicole" and an entity page about the capital city of France for "Paris".

Another limitation is that Stanford NER, and consequently also our system, does not distinguish between real-world and fictive people. This is problematic because fictive people is not affiliated with person-specific temporal data, such as date of birth or death. Additionally, date of birth is not guaranteed to only contain just one value⁶. Nor is it guaranteed that all person entities are affiliated with a date of birth, as there are no requirements for a person entity page to contain an infobox or which parts of the infobox to include. Thus, we cannot assume to always have temporal information about an entity. For cases where the temporal data is unavailable, we have suggested to use cosine similarity for comparing DBpedia entity pages for potential target entities with the document of which the entity mention resides.

Moreover, our system does not consider entity coherence, i.e. it is reasonable to simultaneously assign "Paris" and "France" to the French capital and European country.

⁵The representation used by Wikipedia and DBpedia to indicate a disambiguation page

⁶Observed in at least one person entity: dbpedia.org/page/William_Shakespeare

Our system does not take into account techniques that regards actual alterations of text. This implies that cases of misspellings, words that falsely has been identified as person entities, and words that should have been tagged as person entities, but were not, could be overlooked. This could in part be counter-measured, for example by requiring that only a given set of characters are allowed for person entities. This would cover most of the person entities, and for special cases, such as person entities known by artist names containing unusual characters, e.g. 2pac and 50 Cent, the system would require a few calibrations. However, it does not consider that some data sources are inconsistent when they refer to entities, e.g. a blog post may mention people or organizations using lowercase letters, which would have resulted in that these entity mentions were not recognized by the Stanford NER.

5.7 Evaluation of Experiment Metrics and Results

The experiments revealed significant fluctuations in the distribution of HTTP responses, which consequently had an impact on the precision score. Although the intersection of entity mentions with successful HTTP responses across the experiments more than doubled the precision score compared to the individual experiments, an entity linking system with a precision score of 28 % is not competitive when compared to state-of-the-art systems. In comparison, Stanford's contribution in TAC KBP 2014 gave an overall result of 54.6 % precision [7], entity linking on microblogs with spatial and temporal signals had a precision score of 85.5 % [32], and the entity linking system TAGME had up to 90 % precision [33]. However, as we have only been able to calculate the precision score, and not recall nor the F_1 -score, these numbers does not provide us with a good enough basis to evaluate our system by comparing it to other systems.

The reason for this is that the precision score only conveys one aspect of relevance, namely the exactness of the classification, while we lack a measure to express the completeness of the classification, i.e. recall. This is pertinent because recall and precision are inversely related, and because of this relationship, a moderately good performance on the harmonic weighted mean of these two measures is more interesting than an great performance for precision and a terrible performance for recall, and vice versa. It is also important to consider that all of the aforementioned measures assumes that we know how many person entities there are in the document corpus we process in our experiments, but as explained in section 5.1, we cannot be entirely sure that the entity mentions our system has retrieved corresponds to the

actual number of person entities that exists in the data.

Another challenge when it comes to comparing our system to that of others, it that it is almost impossible to assess entity linking approaches. This is because, if we wish to compare the performance system A to the performance of a reference system B, both systems must either process the exact same dataset, or system B must be reimplemented, and run on the same dataset as system A. This depends on the repeatability of system, which means that system A should be implemented under the same conditions as the reference system [39]. This is challenging because the datasets and applied technology may no longer be available, as we ourselves experienced during the course of this project (see section 3.6.4), the documentation may be unavailable, outdated or incomplete, or the implemented system may diverge from the descriptions in the documentation, which makes it difficult to reproduce results, and thus to perform a correct evaluation.

Furthermore, cosine similarity was not used to compare DBpedia entity pages of potential target entities with a document wherein an ambiguous entity mention resides. This was because the temporal expressions identified in the document corpus were largely implicit and relative temporal expressions, which would have required some sort of transformation into an explicit temporal expression to be able to map the temporal expression to a temporal data property in DBpedia. Moreover, the temporal data attributes in DBpedia only were a few data points that represented important life events for the person entity, which made it difficult to find direct matches in the content-based temporal expressions. As we suggested to use cosine similarity for cases where temporal data was unavailable, this would in effect entail that all the entity mentions and corresponding potential target entities would have to be measured with cosine similarity, defeating the purpose of regarding the temporal aspect of the data.

5.8 Research Questions Revisited

In the introduction of this thesis, we stated that our goal was to investigate whether it was possible to improve named entity disambiguation by considering the temporal aspect of the data. In that regard, it is interesting to review how others have approached the problem, both when it comes to exploiting temporal data, for the polysemous aspect of the named entity disambiguation problem, and to reuse some of the knowledge others have learned through their work.

Our first research question touched upon what kind of temporal data about entities it is possible to obtain. As we have described in section 2.6, there are several types of temporal data, including explicit, implicit and relative temporal expressions, which could be both content-based and non-content-based. Combined with the temporal data available in e.g. Wikipedia infoboxes and DBpedia, this provided us with the necessary means to explore whether this temporal information could be exploited to disambiguate ambiguous entities. This was approached by preprocessing and parsing a document corpus, and then use an information extraction framework to extract person entities and temporal entities from the text. The extracted person entities were then used in a HTTP request to check whether the entity in its current form returned a successful HTTP response code.

To exploit temporal data for named entity disambiguation, we could then compare the available temporal data for each of the entity mentions returning a successful HTTP response in the knowledge base with the normalized temporal entities. This brings us to our last research question, which is whether it is expedient to use temporal data for named entity disambiguation. The answer is: It could be both useful and practicable, but it is also likely inexpedient because named entity disambiguation based on temporal data depends on there being explicit temporal expressions in the content, as well as accurate document metadata. This is problematic because we cannot guarantee the accuracy of the document metadata without further analysis, and although temporal expressions may occur in the content, they are improbable to be explicit temporal expressions because humans are unlikely to be explicit and unambiguous when referring to temporal expressions. Thus, should there be non-explicit temporal expressions in the content, they would have to be transformed into explicit expressions in order to link them to the temporal data in the knowledge base, which only contained a few data points representing important life events for the person entity. Transforming non-explicit temporal expressions into explicit temporal expressions would require a lot of resources. Thus, the scarcity of explicit content-based temporal data combined with the limited temporal data in the knowledge base is likely to outweigh the potential useful information that could have been used for named entity disambiguation.

6.1 Conclusion

The named entity disambiguation problem poses a challenge in several research fields. In this thesis we have investigated whether it was possible to exploit temporal data to improve named entity disambiguation, and ultimately bring us one step closer to the solution of the named entity disambiguation problem.

Our proposal was to perform entity linking to check whether mentions of person entities identified in a semantically and syntactically tagged document corpus could be linked directly to its corresponding entity in DBpedia. We examined what kind of temporal data that could exist about person entities, both in the content and metadata of a StreamItem, as well as the temporal attributes in DBpedia. We initially planned to use precision, recall and F_1 -score to evaluate our entity linking approach, and suggested using cosine similarity to compare the similarity of the potential target entity pages in DBpedia with the content of the StreamItem wherein the entity mention resided for cases where the temporal data was unavailable. Unfortunately, we were unable to calculate recall, and thus nor the F_1 -score, as we could not be certain that all the person entities in the corpus had been extracted, which made it impracticable to make a reasonable comparison of our results with results from state-of-the-art systems. Nor did we calculate cosine similarity, as we were unable to detect specific temporal data properties in DBpedia that could have been used for named entity disambiguation given the content-based and non-content-based temporal data that was available to us through the subset of the document corpus we processed.

There has been no indications that the generation of online data will decelerate in the years to come, and a solution to the polysemous aspect of the named entity disambiguation problem becomes key in fully exploiting the potential invaluable

information hidden in this data. The challenges affiliated with the named entity disambiguation problem that has been discussed in this thesis is a pertinent contribution to research on the named entity disambiguation problem, and forms a solid basis for general and further work with the polysemous aspect of named entity disambiguation problem.

6.2 Future work

One of the challenges we encountered during the development of the system is that common nouns are not considered candidates for named entities, as described in section 3.6.1, which was a restriction set by the NER annotator in the Stanford CoreNLP parser. Moreover, limitations in resources forced us to diverge from our original plan, which was to use three parsers rather than one, as this was likely to boost system accuracy. For future improvements of our system we therefore suggest to combine the Stanford CoreNLP parser with at least two other parsers, and to increase the focus on part-of-speech tagging, phrase structure parsing and dependency parsing for identifying named entities that currently is recognized as regular nouns by the Stanford NER. Moreover, we suggest to implement methods that handles misspellings and errors that was caused by special characters. Although distant supervision was futile in our case, we do recognize its potential in NLP research, and believe it may be expedient in future work, for example by generating a dataset consisting solely of person entities by extracting person entities from a knowledge base, and use this dataset to test the entity linking performance by comparing it to a dataset containing several types of entities. This could also be expanded or changed to include other types of entities, given that they belong to a concept that is defined by the knowledge base.

As described in section 4.1.2, the size of the document corpus had to be confined due to restrictions imposed by time and resources. If not impeded by these factors however, processing the full dataset would have been preferable as it for example could have revealed interesting patterns that was not apparent in the fraction of the dataset that was processed. Our entity linking approach would be possible to scale to fit the full dataset, but would be impracticable because the semantic and syntactic annotation alone would have taken years to complete if it was only run on one machine. We therefore propose that a system like the one we describe in this thesis would benefit from parallel, distributed processing over multiple machines, which would be available through programming models such as MapReduce. Enforcing such a programming model would allow the operations to be run in parallel on mul-

tiple computers simultaneously, and thus drastically reduce system runtime.

The approach we have described in this thesis is directed towards temporal data, but as temporal and spatial data in this context largely is represented using the same backbone, the system could be easily adapted to fit spatial data in addition to temporal data. For example, the Stanford NER identifies spatial entities as well as temporal entities, and DBpedia provide attributes pertaining to locations in addition to the temporal attributes we described in section 3.2.2. This may an interesting aspect to pursue in future work, especially considering the spatial data increasingly becoming available in social media, e.g. Facebook allow users to tag their location, and in Twitter, all Tweets are timestamped, and in many cases also mapped to a location [32].

The document corpus processed in this thesis is static offline data, which is appropriate for our purposes because we want the data to be stable during experimentation to be able to determine whether any progress had been made regarding entity linking quality and named entity disambiguation. However, once improvements have been made and the system have been stabilized, it may be interesting to adapt the system to accommodate for continuous data streams. Given the same preconditions, it may also be interesting to expand the system to accommodate for multiple languages. A language-independent named entity disambiguation incorporated in research areas such as KBA and KBP could create a low threshold for third world countries to generate knowledge bases in their own language, and stimulate to easier access to education and vast amounts of knowledge.

APPENDIX A

Experiment Results

1	$\frac{42.605}{294.403} = 14.471 \%$	8	$\frac{43.040}{294.403} = 14.619 \%$	15	$\frac{43.062}{294.403} = 14.626 \%$
2	$\frac{42.512}{294.403} = 14.440 \%$	9	$\frac{42.112}{294.403} = 14.304 \%$	16	$\frac{43.990}{294.403} = 14.942 \%$
3	$\frac{42.948}{294.403} = 14.588 \%$	10	$\frac{41.774}{294.403} = 14.189 \%$	17	$\frac{43.162}{294.403} = 14.566 \%$
4	$\frac{36.114}{294.403} = 12.267 \%$	11	$\frac{47.937}{294.403} = 16.282 \%$	18	$\frac{43.217}{294.403} = 14.434 \%$
5	$\frac{43.181}{294.403} = 14.667 \%$	12	$\frac{45.347}{294.403} = 15.403 \%$	19	$\frac{42.679}{294.403} = 14.254 \%$
6	$\frac{43.977}{294.403} = 14.937 \%$	13	$\frac{43.905}{294.403} = 14.913 \%$	20	$\frac{43.879}{294.403} = 14.655 \%$
7	$\frac{43.713}{294.403} = 14.848 \%$	14	$\frac{43.564}{294.403} = 14.797 \%$		

Table A.1: Precision score for when TP equals successful HTTP requests

#	000	200	301	404	500	502	503
1	43	42.605	12.590	90.251	131	-	148.783
2	44	42.512	12.555	90.106	124	-	149.062
3	25	42.948	12.733	90.878	126	-	147.693
4	117	36.114	10.668	76.604	95	21.171	149.634
5	35	43.181	12.762	91.527	129	-	146.769
6	62	43.977	13.158	92.729	115	-	143.241
7	114	43.713	12.967	92.176	129	-	145.304
8	33	43.040	12.698	90.992	110	-	147.530
9	-	42.112	12.625	89.586	132	-	149.948
10	-	41.774	12.371	88.302	120	-	151.836
11	13.482	47.937	14.196	101.602	140	-	117.046
12	54	45.347	13.484	95.445	127	177	139.769
13	333	43.905	12.913	92.871	135	-	144.246
14	18	43.564	12.868	92.646	139	-	145.168
15	85	43.062	12.771	91.722	118	-	146.645
16	45	43.990	13.017	92.615	120	127	144.489
17	27	43.162	12.930	91.499	123	-	146.662
18	35	43.217	12.757	91.548	131	-	146.715
19	39	42.679	12.705	90.538	118	-	148.324
20	271	43.879	12.874	92.898	122	-	144.359

Table A.2: Distribution of HTTP response status codes for the experiments

APPENDIX B

Parser annotators

This thesis used the Bazaar parser¹ to perform annotations. The Bazaar parser is a wrapper of the Stanford CoreNLP parser, and is the embedded parser of DeepDive. Table B.1 includes a brief description of each of the annotators used in this thesis².

PROPERTY	DESCRIPTION	DEPENDS ON
tokenize	Tokenizes the text. Originally based on the Penn Treebank standard, but later extended to handle noise and web text	None
cleanxml	Remove XML tokens from the document	tokenize
ssplit	Splits a sequence of tokens into sentences	tokenize
pos	Labels tokens with their Part-of-speech tag	tokenize, ssplit
lemma	Generates the base words (lemmas) for all tokens in the corpus	tokenize, ssplit, pos
ner	Recognizes named (PERSON, LOCATION, ORGANIZATION, MISC), numerical (MONEY, NUMBER, ORDINAL, PERCENT), and temporal (DATE, TIME, DURATION, SET) entities	tokenize, ssplit, pos, lemma

Table B.1: Parser annotators

¹[HTTPS://github.com/HazyResearch/bazaar](https://github.com/HazyResearch/bazaar)

²A full overview of annotators, dependencies and more details is available at [HTTP://stanfordnlp.github.io/CoreNLP/annotators.html](http://stanfordnlp.github.io/CoreNLP/annotators.html)

Bibliography

- [1] P. Eston, *Text Analytics with Ambiverse*, ch. 2, pp. 7–11. ambiverse.com, 1 ed., 2016.
- [2] M.-C. De Marneffe, B. MacCartney, C. D. Manning, *et al.*, “Generating typed dependency parses from phrase structure parses,” in *Proceedings of LREC*, vol. 6, pp. 449–454, 2006.
- [3] O. Alonso, J. Strötgen, R. A. Baeza-Yates, and M. Gertz, “Temporal information retrieval: Challenges and opportunities,” *TWAW*, vol. 11, pp. 1–8, 2011.
- [4] C. Zhang, *DeepDive: A Data Management System for Automatic Knowledge Base Construction*. PhD thesis, UW-Madison, 2015.
- [5] G. Angeli, S. Gupta, M. J. Premkumar, C. D. Manning, C. Ré, J. Tibshirani, J. Y. Wu, S. Wu, and C. Zhang, “Stanford’s Distantly Supervised Slot Filling Systems for KBP 2014,” in *Proceedings of the Seventh Text Analysis Conference (TAC 2014)*, National Institute of Standards and Technology (NIST), 2015.
- [6] A. Dragland, “Big data—for better or worse,” *SINTEF*, vol. 22, May 2013.
- [7] M. Surdeanu and H. Ji, “Overview of the English Slot Filling Track at the TAC2014 Knowledge Base Population Evaluation,” in *Proc. Text Analysis Conference (TAC2014)*, 2014.
- [8] Alan M. Turing, “Intelligent Machinery,” 1948.
- [9] W. Weaver, “Translation,” 1949.
- [10] N. Chomsky, *Syntactic structures*. Mouton & Co, first ed., 1957.
- [11] R. C. Schank, *A conceptual dependency representation for a computer-oriented semantics*. PhD thesis, 1969.

- [12] J. Hutchins, “The first public demonstration of machine translation: the georgetown-ibm system, 7th january 1954,” *Publicación electrónica en:hutchinsweb.me.uk/GUIBM-2005.pdf*, 2005.
- [13] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*, vol. 999. MIT Press, 1999.
- [14] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenu, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum, “Robust Disambiguation of Named Entities in Text,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 782–792, Association for Computational Linguistics, 2011.
- [15] D. Milne and I. H. Witten, “Learning to link with Wikipedia,” in *Proceedings of the 17th ACM conference on Information and knowledge management*, pp. 509–518, ACM, 2008.
- [16] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti, “Collective annotation of Wikipedia entities in web text,” in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 457–466, ACM, 2009.
- [17] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, Volume 1*, pp. 173–180, Association for Computational Linguistics, 2003.
- [18] M.-C. De Marneffe and C. D. Manning, “Stanford typed dependencies manual,” tech. rep., Technical report, Stanford University, 2008.
- [19] J. R. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pp. 363–370, Association for Computational Linguistics, 2005.
- [20] A. X. Chang and C. D. Manning, “SUTime: A library for recognizing and normalizing time expressions.,” in *LREC*, pp. 3735–3740, 2012.
- [21] F. Schilder and C. Habel, “Temporal Information Extraction for Temporal Question Answering.,” in *New Directions in Question Answering*, pp. 35–44, 2003.

- [22] N. Kanhabua, R. Blanco, and K. Nørvåg, *Temporal Information Retrieval*, vol. 9, ch. 3 and 4, pp. 91–208. 2015.
- [23] T. W. Group *et al.*, “Guidelines for Temporal Expression Annotation for English for TempEval 2010,” 2009.
- [24] J. Pustejovsky, J. M. Castano, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev, “TimeML: Robust specification of event and temporal expressions in text,” *New directions in question answering*, vol. 3, pp. 28–34, 2003.
- [25] I. Mani and G. Wilson, “Robust temporal processing of news,” in *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 69–76, Association for Computational Linguistics, 2000.
- [26] S. Nunes, C. Ribeiro, and G. David, “Using neighbors to date web documents,” in *Proceedings of the 9th annual ACM international workshop on Web information and data management*, pp. 129–136, ACM, 2007.
- [27] Z. Zheng, X. Si, F. Li, E. Y. Chang, and X. Zhu, “Entity Disambiguation with Freebase,” in *Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 01, WI-IAT ’12*, pp. 82–89, IEEE Computer Society, 2012.
- [28] G. Angeli, J. Tibshirani, J. Y. Wu, and C. D. Manning, “Combining distant and partial supervision for relation extraction,” in *The 2014 conference on Empirical Methods on Natural Language Processing (EMNLP)*, 2014.
- [29] J. R. Frank, M. Kleiman-Weiner, D. A. Roberts, F. Niu, C. Zhang, C. Ré, and I. Soboroff, “Building an entity-centric stream filtering test collection for TREC 2012,” tech. rep., DTIC Document, 2012.
- [30] K. Balog, H. Ramampiaro, N. Takhirov, and K. Nørvåg, “Multi-step classification approaches to cumulative citation recommendation,” in *Proceedings of the 10th Conference on Open Research Areas in Information Retrieval*, pp. 121–128, LE CENTRE DE HAUTES ETUDES INTERNATIONALES D’INFORMATIQUE DOCUMENTAIRE, 2013.
- [31] R. Abbes, K. Pinel-Sauvagnat, N. Hernandez, and M. Boughanem, “Leveraging temporal expressions to filter vital documents related to an entity,” in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 1093–1098, ACM, 2015.

- [32] Y. Fang and M.-W. Chang, “Entity linking on microblogs with spatial and temporal signals,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 259–272, 2014.
- [33] P. Ferragina and U. Scaiella, “TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities),” in *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1625–1628, ACM, 2010.
- [34] S. Guo, M.-W. Chang, and E. Kiciman, “To Link or Not to Link? A Study on End-to-End Tweet Entity Linking.,” in *HLT-NAACL*, pp. 1020–1030, 2013.
- [35] S. Cucerzan, “Large-Scale Named Entity Disambiguation Based on Wikipedia Data.,” in *EMNLP-CoNLL*, vol. 7, pp. 708–716, 2007.
- [36] R. Bunescu and M. P. a, “Using encyclopedic knowledge for named entity disambiguation,” in *EACL*, pp. 9–16, 2006.
- [37] R. Mihalcea and A. Csomai, “Wikify!: Linking Documents to Encyclopedic Knowledge,” in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, CIKM ’07*, pp. 233–242, ACM, 2007.
- [38] J. Kazama and K. Torisawa, “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*, pp. 698–707, 2007.
- [39] F. Hasibi, K. Balog, and S. E. Bratsberg, “On the Reproducibility of the TAGME Entity Linking System,” in *Advances in Information Retrieval*, pp. 436–449, Springer, 2016.
- [40] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, “Knowledge vault: A web-scale approach to probabilistic knowledge fusion,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 601–610, ACM, 2014.
- [41] O. Deshpande, D. S. Lamba, M. Tourn, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan, “Building, maintaining, and using knowledge bases: a report from the trenches,” in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 1209–1220, ACM, 2013.

- [42] M. Curtiss, I. Becker, T. Bosman, S. Doroshenko, L. Grijincu, T. Jackson, S. Kunnatur, S. Lassen, P. Pronin, S. Sankar, *et al.*, “Unicorn: A system for searching the social graph,” *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1150–1161, 2013.
- [43] F. Mahdisoltani, J. Biega, and F. Suchanek, “YAGO3: A Knowledge Base from Multilingual Wikipedias,” in *7th Biennial Conference on Innovative Data Systems Research*, CIDR Conference, 2014.
- [44] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morse, P. van Kleef, S. Auer, *et al.*, “DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia,” *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
- [45] M. A. Rodriguez, “A Graph Analysis of the Linked Data Cloud,” *arXiv preprint arXiv:0903.0194*, 2009.
- [46] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, *DBpedia: A nucleus for a web of open data*. Springer, 2007.
- [47] Wikipedia:Article_titles. en.wikipedia.org/wiki/Wikipedia:Article_titles. Accessed: 16th September, 2015.
- [48] Wikipedia:Redirect. en.wikipedia.org/wiki/Wikipedia:Redirect. Accessed: 15th September, 2015.
- [49] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [50] R. Baeza-Yates, B. Ribeiro-Neto, *et al.*, *Modern Information Retrieval*, vol. 463. ACM press New York, 1999.
- [51] Wikimedia, “Stop word list/google stop word list.” meta.wikimedia.org/wiki/Stop_word_list/google_stop_word_list#English, April 2013. [Online; accessed 22d May 2016].
- [52] D. Harman, “How effective is suffixing?,” *Journal of the American Society for Information Science*, vol. 42, no. 1, p. 7, 1991.
- [53] W. B. Frakes and R. Baeza-Yates, *Information retrieval: Data structures and algorithms*, ch. 8. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.

- [54] S. S. Pradhan, L. Ramshaw, R. Weischedel, J. MacBride, and L. Micciulla, “Unrestricted coreference: Identifying entities and events in OntoNotes,” in *null*, pp. 446–453, IEEE, 2007.
- [55] E. Boschee, P. Natarajan, and R. Weischedel, “Automatic extraction of events from open source text for predictive forecasting,” in *Handbook of Computational Approaches to Counterterrorism*, pp. 51–67, Springer, 2013.
- [56] J. Dalton, J. R. Frank, E. Gabrilovich, M. Ringgaard and A. Subramanya, “FAKBA1: Freebase annotation of TREC KBA Stream Corpus, Version 1 (Release date 2015-01-26, Format version 1, Correction level 0),” January 2015. Data available at <http://trec-kba.org/data/fakba1/>[Accessed: 8th March, 2016].
- [57] E. Boschee, R. Weischedel, and A. Zamanian, “Automatic Information Extraction,” in *Proceedings of the 2005 International Conference on Intelligence Analysis*, McLean, VA, pp. 2–4, Citeseer, 2005.