Automatic Word-Sense Disambiguation Using Inherited Hypernyms

Julia Fellner

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2015

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 18, 2015

Julia Fellner

Contents

Declaration iii

Kurzfassung vii

Abstract viii

1 Introduction 1
- 1.1 Problem Statement . . . 1
- 1.2 Goal . . . 2
- 1.3 Structure . . . 2

2 Technical Background 3
- 2.1 Natural Language Processing . . . 3
- 2.2 Information Retrieval . . . 4
- 2.3 Word-Sense Disambiguation . . . 5
- 2.4 Artificial Intelligence . . . 5
- 2.5 Vector Space Model . . . 6
- 2.6 Machine Learning . . . 6

3 State of the Art 7
- 3.1 Semantic Web . . . 7
  - 3.1.1 Vocabularies for structured data . . . 8
- 3.2 Word-Sense Disambiguation . . . 11
  - 3.2.1 Categories . . . 12
  - 3.2.2 Automatic Approaches to Word-Sense Disambiguation 13
- 3.3 Schema.org . . . 15
  - 3.3.1 Terminology . . . 15
  - 3.3.2 Statistics . . . 18
- 3.4 Information Retrieval APIs . . . 19
  - 3.4.1 Alchemy API . . . 20
  - 3.4.2 AYLIEN API . . . 20

4 New Approach 23
- 4.1 Concept and Architecture . . . 23

4.1.1 Goal . . . 23
4.1.2 System Design . . . 24
4.2 Information Retrieval . . . 24
4.2.1 Dictionaries and Inherited Hypernyms . . . 25
4.2.2 Wikis . . . 26
4.2.3 Search Engine Information Retrieval . . . 27
4.2.4 Entity Type Sub Pages . . . 27
4.3 Single Entity Detection . . . 28
4.3.1 Differentiation by Properties . . . 29
4.3.2 Differentiation by Context . . . 29
4.3.3 Term Frequency and Inverse Document Frequency . . . 30
4.3.4 Single Entity Identifiers . . . 31
4.4 Data Comparison and Confidence Scoring . . . 32
4.4.1 Group Entity Identifiers . . . 33
4.4.2 Inheritance Similarity . . . 34

5 Implementation 36
5.1 Framework . . . 36
5.1.1 NodeJS . . . 37
5.1.2 Node Modules . . . 37
5.2 Database . . . 38
5.2.1 MongoDB . . . 38
5.3 Application Structure . . . 39
5.3.1 Models and Data . . . 40
5.3.2 Updates . . . 40
5.3.3 Preprocessing . . . 42
5.4 Text Analysis . . . 47
5.4.1 Filtering . . . 47
5.4.2 Single and Group Identifiers . . . 48
5.4.3 Synonyms and Stemming . . . 49

6 Evaluation 50
6.1 Test data collection . . . 50
6.1.1 Test Example Model . . . 51
6.1.2 Expected Test Result . . . 52
6.2 Testing environment . . . 53
6.2.1 AYLIEN's Text Analysis API . . . 53
6.2.2 Test Query Representation . . . 54
6.3 Testing Categories . . . 54
6.3.1 Stemmed Group Test . . . 54
6.3.2 Lexical Knowledge Base Test . . . 55
6.3.3 Single Entity Test . . . 56
6.4 Results . . . 56
6.4.1 Lexical Knowledge Base Test Result . . . 57

6.4.2 Stemmed Group Test Result 59
6.4.3 Single Entity Test Result 60
6.5 Overall Evaluation 61
6.5.1 Specific Vocabulary and Entity Observations 61

7 Conclusion **63**
7.1 Outlook 64

A Contents of the DVD-ROM **65**
A.1 PDF files 65
A.2 Source Code 65
A.3 Other 66

References **67**
Literature 67
Online sources 69

Kurzfassung

Nutzer aus aller Welt stellen in der heutigen Zeit eine Flut an Daten ins Netz und machen es dadurch reicher, was aber zur Folge hat, dass im Web in einem rasenden Tempo immer mehr unstrukturierte Informationen unterwegs sind. Die Suchmaschinen sind stark gefordert, vor allem wenn es darum geht, schnell und effizient zu arbeiten, sprich das Richtige zu finden.

Überwachte semantische Systeme nutzen Wortsammlungen mit deren Synonymen und diversen Ausdrucksweisen, um das Problem der Mehrdeutigkeit eines Wortes beim automatischen Einfügen von Informationen eines Bedeutungsfelds zu umgehen. Verwandte Begriffe werden dazu mit ihren Wort-Beziehungen in großen Daten-Kollektionen gespeichert. Nicht bzw. wenig überwachte Systeme haben allerdings noch Schwierigkeiten semantische Entitäten einem Wortsinn entsprechend zuzuordnen, da diese auf das Sammeln und Speichern dieser Relationen zwischen Wörtern verzichten, um flexibler und schneller zu handeln. In dieser Arbeit wird die Idee zur Nutzung strukturierter Vererbung und verwandter Oberbegriffe dargelegt, um das übergreifende Konzept eines Suchbegriffs automatisch zu bestimmen. Semantische Vokabulare werden mit diversen Wissensdatenbanken und Wörterbüchern verknüpft, um die Basis des selbständigen Systems zu bilden. Dieser Schritt ermöglicht die Reduzierung und etwaige Eliminierung einer zusätzlichen Datenbank zur Beschreibung von Vokabular-Entitäten. In Folge dessen werden Webseiten semantisch aufbereitet und bereichert, während das unstrukturierte World Wide Web zum Web der Daten transformiert wird.

Abstract

Through the vast amount of user-generated content from people around the world the web has become a rich database, filled with unstructured data thanks to social networking websites and collaborative knowledge bases. It is challenging for machines, such as search engines, to rapidly find and understand such content correctly.

Supervised semantic approaches in this area use database collections to eliminate the problem of identifying a word's correct meaning when trying to automatically include semantics into unstructured content. These systems use large data-collections to save related terms and synonyms for this process. Unsupervised and semi-supervised systems on the other hand still struggle with the challenge of word-sense disambiguation when wanting to include semantic entities into plain text. Via the introduction of inherited hypernyms, representing structural inheritance of semantic entities within vocabularies, and hierarchical information, the identification of a word's true meaning, by determining their overall topic and concept, is described in this thesis. Semantic vocabularies are combined with various knowledge bases to build the base of the automatic system and find the best suitable entity for a specific term in a semi-supervised manner. This allows the system to reduce the size of an optional word-relations data collection, enrich a website's content semantically and support the transformation of the World Wide Web into a Web of Data.

Chapter 1

Introduction

In many natural languages a word or term can represent multiple meanings and senses. Word-sense disambiguation (WSD) is one of the fundamental tasks in natural language processing (NLP). A given ambiguous word, such as "Avatar", is to be determined in its sense of word to being either a known movie or an image representation of a profile. WSD is a known problem in Computational Linguistics, influencing various applications for machine translation, information extraction and information retrieval. The most commonly used technique to solve such a problem is to use the surrounding context of an ambiguous term in a probabilistic calculation.

The World Wide Web as we know it today is a supersaturated content network, full of data without structure. Texts, images and videos are created online on a daily basis, increasing this amount of unstructured information rapidly. The difficulties to understand and interpret such unstructured text to identify valuable terms and their correct meaning is challenging. Semantic vocabularies, including markup structures, represent tools to incorporate machine-readable information into web pages. In combination with semi- and unsupervised systems, they provide the solution to transform the unstructured web and its fast growing content into a Web of Data. But the so far published approaches in automatic text analysis need to overcome the well-known problem of WSD.

1.1 Problem Statement

Any kind of data without semantic markup in form of attributes or descriptive properties is considered to be unstructured content. For a machine to understand such plain text, valuable terms need to be extracted and their semantically correct entity types are to be identified. Various approaches for automatic semi- and unsupervised WSD using lexical and collaborative knowledge bases have been presented with a maximum precision value of 69% (SemEval 2007 [17]) correctly identified terms. That result leaves room

for improvement and the development of a system to automatically determine a word's true meaning using revised methods and heuristics. While the issue of word-sense disambiguation has been dealt with by using collaborative and lexical knowledge bases with their definitions and descriptions of terms [9, 21], there are no techniques available that compare inherited hypernyms of semantic vocabularies and knowledge bases.

1.2 Goal

Its intelligent and independent behaviour is the overall goal of the system's design. In order to overcome the problem of WSD and eliminating it at an earlier stage of text analysis, a new approach is presented including inherited hypernyms in combination with lexical and hierarchical structure information. The goal is set to investigate the possibilities and restrictions of automatic systems in the field of semantic analysis, as well as to approach and potentially surpass the achieved precision value in SemEval 2007 [17] for identifying a word's true meaning. After the inclusion of already defined entities from an existing semantic vocabulary, the system has to operate intelligently based on the decision structure which is implemented to determine the correct entity types of various terms. No data-set and word database has to be created, to comply with the predefined independence of an automatic system.

1.3 Structure

This document is divided into four main chapters to state, analyse and evaluate a developed system for automatic word-sense disambiguation using inherited hypernyms. Chapter 2 provides short definitions of various technical terms in the fields of natural language processing and machine learning, which represent a necessity for the understanding of this thesis. In Chapter 3 the basic terminologies for the fields of Semantic Web and WSD are described, as well as some fundamental technical background information of the research field. Chapter 4 presents the proposed automatic system with its design. The main approach with its new combination of heuristics and lexical information to eliminate the problem of word-sense disambiguation in an early stage is described in detail. How the system design and main idea is realised in development, is described in Chapter 5. Here, the used framework, database and application structure with all their necessary technologies to realise the new approach are presented. Finally, in Chapter 6 the environment for testing the implemented system is described and the results are evaluated. Chapter 7 concludes the document describing the outcomes of this work.

Chapter 2

Technical Background

The fields of Natural Language Processing, Information Retrieval and Machine Learning contain various technical terms and acronyms, which are necessary to be known for the understanding of this work. In this chapter the most important ones are shortly explained. Regarding the basic terms of web development used in this paper, the reader is expected to have a certain background knowledge in that area.

2.1 Natural Language Processing

Natural Language Processing (NLP) represents a Computer Science field, which is connected to Artificial Intelligence and Computational Linguistics. It focuses on interactions between computers and human language and a machine's ability to understand, or mimic the understanding of human language (e.g. Siri or Google Now).

In the field of Natural Language Processing various terms are used to describe specific technical approaches which are shortly described in the following sub-paragraphs.

Information Extraction

In the process of Information Extraction (IE) structured information is to be automatically identified and extracted out of semi- and/or unstructured sources (e.g. text documents or web pages). IE represents an important part in the computation of unstructured data and to allow logical reasoning based on its results.

Named Entity Recognition

Named Entity Recognition (NER) is based on predefined categories, such as the names of people, organizations, monetary values, percentages and places. Semantic vocabularies represent a hierarchical structure and the interconnection of and between such categories (or entities). NER describes the categorization and location of plain text into these defined classes.

Corpus or Corpora
A corpus is usually a large collection of documents, which can be used to infer and validate linguistic rules.

Bag of Words
Bag of Words (BOW) represents a commonly used model in methods of Text Classification. A piece of text, which can be a document or a sentence, is represented as a bag or multi-set of words. The word order and grammar have no importance in this process, as the frequency and number of occurrences of each word are primarily used as a feature for training a classifier.

Stop Words
In computing so called *stop words* represent words which are filtered out before or after the processing of natural language data. These words are used quite often in most languages. But despite of their high usage they do not carry a lot of meaning. Basically any group of words can be added to a stop word list for a given purpose. For some search engines, these are some of the most common structure- or function-words that exist, such as *the, it, which, on, as* and *at* in the English language.

2.2 Information Retrieval

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources, such as text documents and websites. To reduce information overload and determine the most important terms within a plain text various calculations and formulas are used.

Term Frequency
The easiest choice and most basic calculation for the Term Frequency (TF) of a word is to use the *raw frequency* of a term in a specific document. For example the number of times term t occurs in document d. A document that mentions a query term more often is thought to have more in common with that query and therefore should receive a higher score. The query is seen as a simple set of words in most search algorithms. To calculate a weight for each term, its occurrences are summed up and the exact order of words is ignored (bag of words model).

Since every document is different in length, it is expected that a term appears more often in long documents than in shorter ones. As a way of normalization the TF is then usually divided by the length of a specific document, which is represented by the total number of terms in that document.

Inverse Document Frequency

The basic search term weighting formula was proposed by Jones on heuristic grounds in 1972 [19] and is denoted

$$f_{\mathrm{id}}(t, D) = \log \frac{|D|}{1 + \mathrm{card}\{d \mid d \in D, t \in d\}}. \tag{2.1}$$

Inverse Document Frequency (IDF) is described as the logarithmically scaled fraction of the documents that contain the term t. The document corpus D represents a collection of single documents d that each include various terms t. To calculate how often a single term of a specific document appears within the document the total number of documents $|D|$ needs to be divided by the number of documents containing the term. The adjustment of adding the value 1 in the denominator is necessary to avoid a division-by-zero (in case the term does not appear in the corpus). At the end, the logarithm of the result value is taken.

Term Frequency – Inverse Document Frequency

Through the multiplication of Term Frequency and Inverse Document Frequency the TF-IDF value, which indicates the importance of a word within a specific document corpus D, is calculated using the following formula:

$$f_{\mathrm{tfid}}(t, d, D) = f_{\mathrm{tf}}(t, d) \cdot f_{\mathrm{id}}(t, D). \tag{2.2}$$

The resulting value of TF-IDF is high when t occurs often within a small number of documents (thus lending high discriminating power to those documents). The number is low when the term occurs only a few times within a document or occurs in many documents (thus offering a less pronounced relevance signal). If the number is zero or just very low, then the term occurs in (almost) every document.

2.3 Word-Sense Disambiguation

The term Word-Sense Disambiguation (WSD) explains the ability to identify the true meaning of a word in a given context in a computational manner. In many use cases to approach the problem of correct WSD, a third party corpus or knowledge base, such as WordNet or Wikipedia, is used for cross-referencing entities. A simple example being, to determine the reference of the term *apple* within a text to the fruit or the company. In Section 3.2 the term is explained in more detail.

2.4 Artificial Intelligence

Artificial Intelligence (AI) describes the capability of intelligent behaviour by computer software. In the concept of natural language, systems are devel-

oped to understand natural human languages and perform useful tasks with the natural languages humans use. AI is deeply connected with processes within machine learning and natural language processing to solve problems in their fields.

2.5 Vector Space Model

Vector space models are often fundamental to a host of Information Retrieval operations ranging from scoring documents on a query, to document classification and document clustering.

2.6 Machine Learning

As a sub-field of Artificial Intelligence and computer science, Machine Learning (ML) describes the design of a system, which is able to make decisions and predictions based on data, and learn from them depending on the outcome. Instead of being explicitly programmed to carry out a certain task, computers act and make data-driven decisions themselves. These programs are designed to learn and improve over time when being exposed to new data (e.g. self-driving cars and speech recognition systems). Supervised and unsupervised machine learning represent two important concepts in the area of ML. A brief description of their purpose and use is included in the following paragraphs next to a short explanation of a tool called Decision Trees.

Supervised Learning
: In Supervised Learning a system is trained to make accurate decisions with the use of a pre-defined dataset collection, when given new data. A well-known example is the training of a sentiment analysis classifier. It builds up a dataset of tagged positive, negative and neutral tweets to determine the sentiment of new incoming tweets based on that collection.

Unsupervised Learning
: Unsupervised Learning automatically analyses a given dataset to identify patterns and relationships within the collection. The analysis of emails and their automatic grouping by topic without any prior knowledge or training is one usage example of unsupervised learning (also known as clustering).

Decision Trees
: Decision Trees represent a support tool for systems to make decisions based on a tree-like graph or model of decisions, knowing their possible consequences (e.g. display an algorithm).

Chapter 3

State of the Art

> *What the Gutenberg press did for the creation and distribution of knowledge, the Semantic Web will do for the creation and reutilization of data. To put it short: Semantic Web is the printing press for data.*
>
> — Tassilo Pellegrini, *Semantic Web Company*

To have structured websites by including semantics to their content describes a concept following the objective to enable users to find, share and combine information more-easily on the web. Semantic vocabularies, such as schema.org, are used to include this machine-readable data into web pages. Search engines crawl through web content looking for data to interpret rapidly. The vast amount of valuable information is mostly unstructured, making it less profitable for machines. *The Semantic Web* allows machines to interpret the provided information on web pages via semantic analysis and respond to complex human requests. The process of including semantics automatically into plain texts is still facing the problem of word-sense disambiguation to identify a word's true meaning and insert the correct machine-readable term, which is described in more detail in Section 3.2.

3.1 Semantic Web

The term *Semantic Web* describes a collaborative movement (led by the *World Wide Web Consortium*[1]) that encourages the inclusion of semantic content in web pages. Such content describes microformat information within websites which is readable by machines. The main goal is to convert the existing web, which is mostly filled with unstructured or semi-structured documents, into a web of data. As a result the information is given well-defined meaning, and therefore better enabling computers and people to

[1] http://www.w3.org/

work in cooperation. Structured data markup benefits a website's content not only by getting discovered in search results but also across search engine properties. But the complexity and amount of time that gets consumed by including such data is one of the biggest reasons why the number of websites with structured data is still rather low [36]. *Web Data Commons* states that only 620 million HTML pages out of 2.01 billion parsed pages contain some sort of structured information.

3.1.1 Vocabularies for structured data

To optimize a web page one can choose between various vocabularies and syntax options for structured data markup to include semantic information into their web content. The most general and common ones are:

- schema.org,
- data-vocabulary.org,
- microformats,
- RDF, RDFa, RDFa Lite,
- microdata and
- JSON-LD.

The launch of schema.org in June 2011 was for many search marketers not only an initiative itself, but also a massive opening to the world of structured data. The project is generally associated with and also derived from a number of technologies that are described in the upcoming paragraphs.

Schema.org

> Schema.org provides a collection of shared vocabularies webmasters can use to mark up their pages in ways that can be understood by the major search engines: Google, Microsoft, Yandex and Yahoo.

The described vocabulary isn't comparable with an intellectual one for humans. In semantics, vocabularies are defined as concepts and relationships (also known as terms) used to represent and describe an area of concern. They are then used to define possible constraints, characterize potential relationships and classify the terms that are used in a particular application. The complexity differs from thousands (also described as ontologies) to just a couple of terms, which are defined within the collection [23]. Schema.org has currently over 600 terms in its vocabulary and the term *schema* itself represents a set of rules and definitions, or more formally: it is a way to define the structure, content, and to some extent, the semantics of XML documents [1, p. 166]. Schema.org shouldn't be confused with descriptions

such as microformat, microdata or markup: it is a markup vocabulary.[2] Schema.org's Aaron Bradley states in a Posting on the social media platform *Google+* a more precise explanation of it and the difference between XML and RDF-based schemas [29]:

> Sometimes we talk like schema.org is one big schema; sometimes as if it were several. This is because it has an associative, network structure. You can see similar ambiguity about how other networks are discussed. The word *vocabulary* emphasises description and communication. The word *schema* emphasises data structures, databases. Unlike XML schemas, RDF-based schemas are closer to dictionaries than to grammar rules. They document the meaning and inter-relationship of descriptive terms rather than police strongly how you must use them.

Data-vocabulary.org

Data-vocabulary.org represents the predecessor to schema.org and it was the primary vocabulary used for marking up HTML documents with microdata before the publication of schema.org. Data-vocabulary.org is still widely deployed and many tool- and information-based web pages on rich snippets, such as the *Google Webmaster Tool*, still include it in their examples. But the developer community around it is not likely to grow with the existence of schema.org.

Microformats

A microformat represents a vocabulary as well as a markup syntax. It is also described as a method of adding semantic information to an HTML document. For that, microformats use a prescribed markup structure that relies on already existing HTML attributes. The following definition of microformats is to be found on the official website:[3]

> Microformats are simple ways to add information to a web page using mostly the class attribute (although sometimes the id, title, rel or rev attributes do too). The class names are semantically rich and describe the data they encapsulate.

The concept of Microformats differentiates itself from other semantic vocabularies in various important ways. It lacks for example schema.org's hierarchical structure, which enables inheritance for properties from parent types and allows the interconnection between entity types. Microformats are individuals and therefore standalone schemas. One of the biggest differences

[2] A more detailed description on schema.org is found in Section 3.3
[3] http://microformats.org/

between microformat schema and schema.org is the way of how they are included in HTML content. While microformats do depend on class attributes to be added, schema.org allows itself to be included into web documents in multiple ways, namely microdata, RDFa and JSON-LD. Therefore, it can be concluded that microformats can't be compared to similar vocabularies, but to other structured data syntaxes. Microformats are also more limited in scope than broader vocabularies and less extensible than for instance schema.org.[4]

RDF, RDFa and RDFa Lite

The Resource Description Framework (RDF) is a standard model for data interchange on the Web and serves as a language for representing information about resources in the World Wide Web. To do so, simple statements, consisting of triples [30] (subject, predicate and object), are used.

RDFa is a RDF markup syntax for structured data. It represents a way of adding semantic information to HTML documents concretely based on RDF. RDFa is very similar in use to schema.org. The main difference in their concepts is that RDFa can be thought of as a RDF syntax, whereas microdata is not, but can be used to extract RDF [29].

RDFa Lite represents, as the name already suggests, a minimal subset of RDFa.[5] The W3C RDFa Lite 1.1 W3C Recommendation describes it as follows [31]:

> The full RDFa syntax ... provides a number of basic and advanced features that enable authors to express fairly complex structured data, such as relationships among people, places, and events in an HTML or XML document. Some of these advanced features may make it difficult for authors, who may not be experts in structured data, to use RDFa. This lighter version of RDFa is a gentler introduction to the world of structured data, intended for authors that want to express fairly simple data in their web pages. The goal is to provide a minimal subset that is easy to learn and will work for 80% of authors doing simple data markup.

Microdata

Microdata[6] is one way to communicate more about a website's content with metadata and supports the markup of structured data in HTML documents. It describes a specific type of information and follows the purpose of helping

[4] http://schema.org/docs/extension.html
[5] http://www.w3.org/TR/rdfa-lite/
[6] http://www.w3.org/TR/microdata/

automated programs to better understand the content of web pages. The concept itself was based on RDFa 1.0, but is now simplified for mainstream web publisher use. The proposed improvements have been adopted in W3C's RDFa 1.1 as RDFa Lite and allow now publishers to use schema.org more-easily alongside other RDF vocabularies. Microdata allows nested groups of name-value pairs to be added to (HTML) documents, in parallel with the existing content. In a section titled "Why microdata? Why not RDFa or microformats?" schema.org states:[7] "RDFa is extensible and very expressive, but the substantial complexity of the language has contributed to slower adoption".

JSON-LD

JSON-LD is defined as a JSON-based format to serialize Linked Data.[8] The recommendation from schema.org to use JSON-LD as a format is relatively new. In summer 2013 the support was announced and examples were added to the schema.org web page for reference in spring 2014. In comparison to the described methods above, which are attribute-based markup, JSON-LD is separated from the existing tags in HTML documents. Kingsley Idehen called JSON-LD: *structured data islands in HTML documents* [29]. The difficulty that occurs for search engines like Google and Bing with this concept of semantic data inclusion is the issue of trust. If they were to accept JSON-LD provided data, it would be possible for spammers to provide malicious structured data in HTML documents hidden behind presentation layers. A solution for this might be digital certificates to enable the trust in data. The interesting thing about JSON as a representational format is the simplicity it provides and the derivation from JavaScript.

3.2 Word-Sense Disambiguation

Homographs are words that represent multiple meanings or senses in many natural languages. Word-sense disambiguation (WSD) describes the process of determining the various meanings a homograph might have in a given context. In some cases it is even difficult for humans to agree on the appropriate sense of a given word within a specific context. Many different classification algorithms have already been used to approach the problem of WSD, such as naive-Bayes, memory-based learners, decision lists and decision trees. Some of the known algorithms are better suited for confronting the WSD problem than others [13, p. 5] and sometimes they are combined to create better results in accuracy.

[7] http://schema.org/docs/faq.html#14
[8] http://www.w3.org/TR/json-ld/

3.2.1 Categories

Approaches in sense disambiguation of words are divided into two main groups: deep and shallow approaches. Whilst deep approaches try to understand the text and are much more difficult to use, shallow ones just consider the surrounding words for information. They are usually implemented with a defined window of n content words. By creating a rule-set and a learning corpus, a computer is trained to automatically derive the sense of a word by analysing and comparing it to its tagged word senses. Deep approaches are in theory more powerful and should have better results for WSD. But with the lack of world knowledge for computers, shallow approaches reach better results in practice. The known techniques for WSD are divided into four main categories: dictionary- and knowledge-based methods, supervised machine learning methods, semi-supervised methods and completely unsupervised methods.

Dictionary- and knowledge-based methods

Dictionary- and knowledge-based methods rely mainly on dictionaries, lexical knowledge bases and thesauri, without using any corpus evidence. The Lesk Algorithm [6] had a big influence in dictionary-based methods. It states that two (or more) words are defined as disambiguated if their description in dictionaries have the greatest word overlap count (Section 3.2.2). A different approach to word overlaps is to consider general word-sense relatedness and to compute semantic similarity of each pair of word senses. This technique is based on a given lexical knowledge base such as WordNet. Graph-based methods are another way of applying WSD and dive into the Artificial Intelligence world of determining word-sense (Section 3.2.2). Knowledge-based methods seek to avoid the need for large amounts of training material, as they try to use pre-existing structured lexical knowledge resources.

Supervised machine learning methods

The focus for supervised machine learning methods is set on the context surrounding the target word. These methods believe that the context can provide enough evidence on its own to disambiguate words. In the included training phase a sense-annotated training corpus is required. Syntactic and semantic features are extracted from the corpus to create a classifier, by using for example a Support Vector Machine [17]. Currently supervised methods perform the best in comparison to other WSD methods with a result of about 80% precision and recall for coarse-grained WSD [14, p. 5]. Although they are able to cope with the high dimensionality of the feature space and wind up with the best results for WSD, they still rely a lot on manual sense-tagging of the corpora for training. This is expensive and cumbersome.

Semi-supervised methods

To overcome the lack of training data, which represents the main problem of supervised methods, semi-supervised or minimally supervised methods make use of a small annotated corpus as seed data [22, p. 3]. That corpus consists of either a small number of sure-fire decision rules or manually tagged training examples. Using any supervised method, the seeds are then used to train an initial classifier. After that each classifier is trained on a much larger training corpus until a given maximum number of iterations is reached or the whole corpus is consumed. Another servant for seed data can be represented by a word-aligned bilingual corpus [16, p. 3].

Completely unsupervised methods

Fully unsupervised methods work from unannotated raw text. They cluster occurrences of words and induce word senses as a result using similarity measures [7]. These sorts of methods are also known as word sense discrimination. A fully unsupervised WSD method using dependency knowledge was presented by Chen et al. in 2009 [3]. Automatically acquired information is often inaccurate and noisy. The results of SemEval 2007 (focusing on unsupervised WSD [17]) show an achievement of about 70% precision and 50% recall. Word-sense disambiguation was included again in SemEval 2013, but with a changed focus on *Cross-Lingual Word-Sense Disambiguation* (CLWSD).

3.2.2 Automatic Approaches to Word-Sense Disambiguation

Word-sense disambiguation is a long-time existing problem in *Computational Linguistics*. Various real-world applications, including machine translation, information extraction and information retrieval, stand under the impact of WSD. Methods commonly deploy the context of a word for its sense disambiguation and the context information may come from annotated text, unannotated text or other knowledge resources. Some examples for such alternative resources would be Extended WordNet [12], Open Mind Word Expert [4], Parallel Texts [16] or Wikipedia for automatic WSD [10]. Over the years many approaches to automatic WSD have been published. In the following subsections the most important ones, in the sense of relatedness to this paper, are presented in more detail.

Automatic Sense Disambiguation using Machine Readable Dictionaries

Michael Lesk published one of the first approaches of automatically detecting sense disambiguation using machine readable dictionaries in 1986 [6]. The so called *Lesk Algorithm* uses machine readable dictionary definitions of

ambiguous words and compares them to the context in which the word occurs. With this procedure Lesk tries to process any text by solely using the immediate context around the target word and the available dictionaries. A synset[9] will then be returned with the highest number of overlapping words between the context sentence and different definitions from each synset. The Simplified Lesk Algorithm (SLA) is frequently used for its simplicity and speed, but has relatively low accuracy. Usually, the context (context window) is local, like the N words or the current sentence around the target word. Auxiliary words, so called stop words [25], are ignored, as they have no lexical meaning. When the algorithm does not find any overlap or several senses have the same overlap score, then the algorithm fails and cannot provide an answer. This causes low recall and defines the main drawback of the SLA [20, pp. 217-227].

In his paper Lesk states that he attempts to create a cheap solution to the problem of sense discrimination by guessing the correct word sense and counting the overlaps between dictionary definitions of various senses. He tried to use an importance value for multiple occurrences and weighting values for the length of dictionary entries. But none of them appeared to have a bigger influence on the results when testing them in practice.

Finding Predominant Word Senses in Untagged Text

A more recent approach for automatic WSD was made public by McCarthy et al. in 2004 [8]. The proposed method performs disambiguation by identifying the most frequent sense of a word in a specific domain using distributional methods. The relatedness of the possible senses of a target word (using Wordnet::Similarity[10]) are measured to a set of words associated with a specific domain. The resulting scores between them are scaled by the distributional similarity score. Raw corpus data is used by this method to automatically find a predominant sense for nouns. The researchers believe that in order to associate the word neighbours in a thesauri with senses they should make use of another notion of similarity, namely semantic similarity. So far methods used the dictionary word neighbours as words themselves and not as senses. The results of their new developed method were evaluated against the hand-tagged resources SemCor and the SENSEVAL-2 English all-words task. The project achieved a precision value of 64% on an all-nouns task, which is 5% lower than the results using first-sense in the manually labelled SemCor. Because of the limited size of hand-tagged resources the researchers believe that an automatic means of finding a predominant sense would be useful for WSD systems. But the idea needs more research and development to achieve improved results compared to hand-tagged resources.

[9] Synset: a set of one or more synonyms [24].

[10] http://wn-similarity.sourceforge.net/

An Experimental Study of Graph Connectivity for Unsupervised Word-Sense Disambiguation

Methods that are based on translational equivalence depend on the difference of word senses in various translations from a source to a target language. Navigli et al. published a paper in 2012 on Multilingual Joint WSD [15] and explain the combination of BabelNet (a large multilingual knowledge base) and a graph-based WSD approach across different languages. The completion of wide-coverage multilingual lexical knowledge resulted in reaching but not beating the state of the art word-sense disambiguation settings. The proposed graph-based WSD algorithm doesn't require sense-annotated data for training but performs significantly better with more incident edges for every node, retrieved for example from WordNet. Navigli et al. focused their experiments primarily on graph connectivity measures and how suitable they are for WordNet-like sense inventories. Therefore, they produced a very generic algorithm for word-sense disambiguation that can be further improved. The team states that they could consider word sequences which are larger than sentences, take syntactic relations into account or score edges in the graph according to semantic importance.

3.3 Schema.org

Google[11], Yahoo![12] and Bing[13], three of the biggest search engines in the world, came together in 2011 to create a common set of schemas for structured data markup on websites called schema.org[14]. Yandex[15], the most popular search engine in Russia, joined the team later on. The project has two main components: an ontology and the expression of this information in machine readable formats. Those formats can be microdata, RDFa Lite and JSON-LD. The ontology describes the vocabulary for naming the types and characteristics of resources of how they are related to each other. Furthermore, it defines the constraints on how to use and describe the given characteristics or relationships.

3.3.1 Terminology

Information that needs to be described follows the terminology of schema.org by being defined as a so-called *item*. In the first step to describe such an item it is to be classified as a specific type of resource. The vocabulary of schema.org is built up as a hierarchy that describes various entities in

[11] https://www.google.com/
[12] https://yahoo.com/
[13] http://www.bing.com/
[14] http://schema.org/
[15] https://www.yandex.ru/

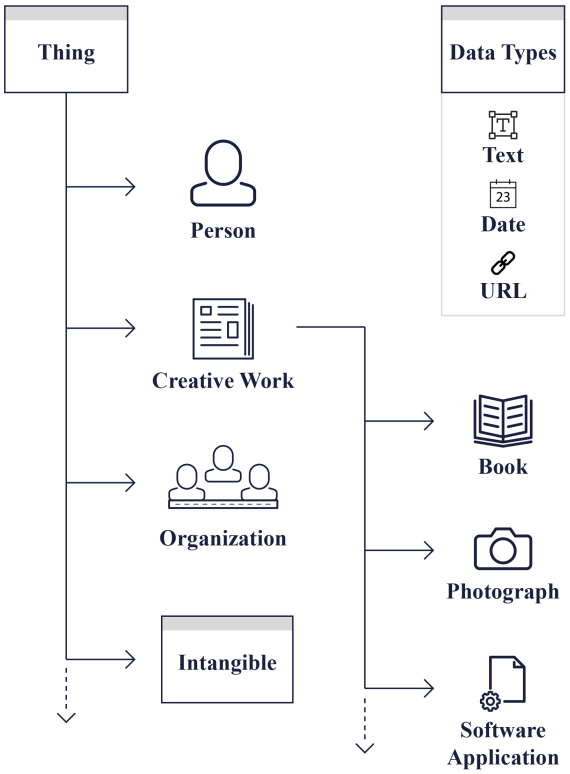


Figure 3.1: Small sample of the schema.org type-hierarchy [38].

different levels of detail. All entities evolve from the main type named *Thing* (Figure 3.1). It represents the most generic entity type. Subtypes of *Thing* include entities such as Person, CreativeWork, Intangible, Organization or Product. The sister parent *DataType* provides descriptive entities such as date, URL or text, that are used to characterize properties of entities. *Thing* acts as the parent node from which all main properties are passed to the child nodes. Every entity type contains the following four attributes to define and identify each single type (given in parenthesis are the expected entity types of the provided property data):

- name (Text): the item's name,
- image (URL): URL of an image of the item,
- url (URL): the item's URL and
- description (Text): a short description of the item.

Almost all entities have children themselves. They inherit their parent's properties, which don't have to but can be included in the mark-up. For

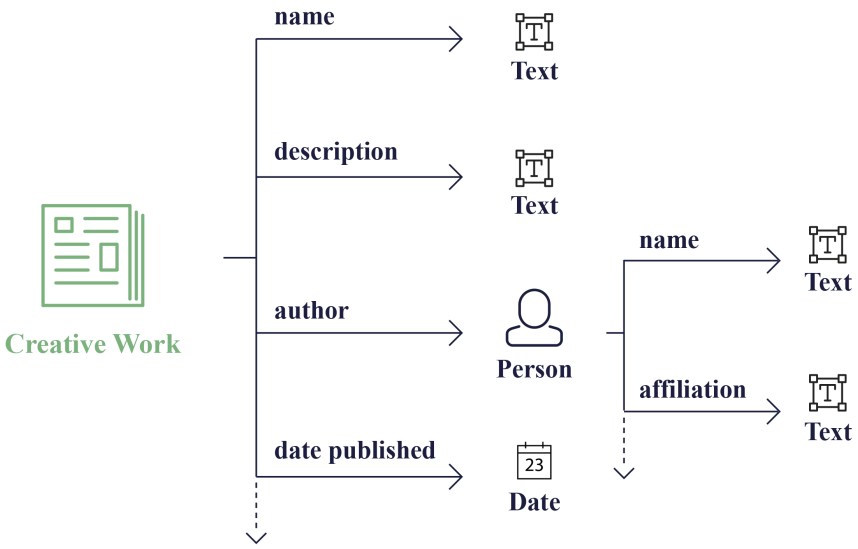


Figure 3.2: Some of the relationships around a Creative Work that may be described using schema.org [38].

example: the entity type *CreativeWork* inherits the given properties of *Thing* and provides around 40 more, including the following:

- about (Thing): the content's subject matter,
- dateCreated (Date): the date on which the CreativeWork was created,
- author (Person or Organization): the content's author and
- publisher (Organization): the creative work's publisher.

Standing out as an entity type in this bullet point list does the property *date* with being a *DataType* entity. It differentiates itself from all other *Thing* properties by being described by an ISO 8601 date format instead of via another entity. The property *author* is, as shown in Figure 3.2, defined through another entity type, which has descriptive properties on its own (linked data connection). Schema.org offers hundreds of terms, but obviously doesn't yet cover many areas in much detail. The community behind *W3C Web Schemas* already discusses ongoing additions and a possible notion of an external enumeration. This should make it clear to how schema.org can be combined with larger vocabularies and datasets from elsewhere (e.g. Freebase, Wikipedia/Wikidata, SKOS).

With an underlying graph-based model of types, various sets of properties and relationships, the schema.org vocabulary represents a very flexible data model which isn't tied to specific file formats. This allows it to be easily communicated, although entity-relationship graphs aren't always the most appropriate data representation. It is clear that no single data format or

Table 3.1: Average website ranking position.

	2013	**2014**
Average ranking position - without Schema	25	25
Average ranking position - with Schema	22	21
Rankings position difference	3	4

abstract data model fully addresses universal needs. But with being so abstract to fit a wide market, it appears to be critical to enter schemas to such a big community. Nevertheless, to use a common vocabulary for semantic markup inclusion, such as schema.org, can have a positive effect on search engine results. The machines will be able to provide richer search results in order for users to find relevant information on the web. Being supported by four of the most popular search engines makes Microdata as a form of semantic data structure more attractive for usage [2, p. 27].

3.3.2 Statistics

SearchMetrics published a study on how schema.org mark-up is used on websites in April 2014 [32]. Over 50 million domains were analysed and 99,70% of them did not have Schema markup on their web page, although the project has been around since 2011. Consequently schema.org mark-up was found in only 0.3% of the remaining domains in the sample. This represents an increase of 0.03% in use of semantic vocabularies compared to the published numbers in 2012 and 2013 [11, p. 3]. That raise is rather minimal but shows still an existing interest of including structured data in web documents. A motivational factor of introducing structured data to websites might also be the improved results in search result rankings. Websites using Schema rank on average four places higher than the ones not including it (Table 3.1).

Bing's web crawler provided a large sample of web data (3,230,928,609), that has been used to present structured data statistics for analysis purposes [11, p. 3]. Around 69% of all URLs did not include any form of semantic information. All other pages incorporated either RDFa (25.08%), microdata (7.16%) and/or microformat (8.6%).

Web data commons published a website crawl and trend analysis on their own. Out of 2.01 billion pages, 620 million HTML pages contained structured data in 2014. The results, as displayed in Figure 3.3, show a significant increase for the inclusion of microdata in websites. The illustration presents the total number of Pay-Level Domains (PLD) [35] making use of either microdata, microformat hCard and RDFa, which represent three of the most widely spread markup formats. *Web data commons* found an increase in deployment of microdata and especially schema.org since the crawl in 2012.

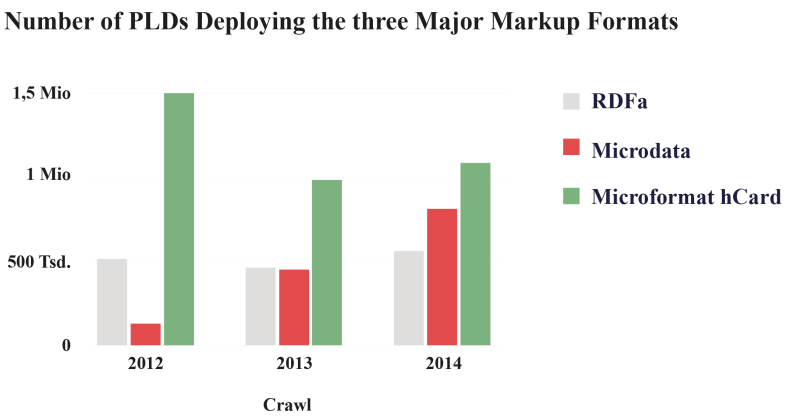


Figure 3.3: Web data commons. Structured data trend 2012–2014 [37].

Schema.org's vocabulary is relatively large, compared to typical RDF vocabularies, as it is designed for mainstream, mass-market adoption. But the project is based on W3C's RDF model for structured data. Schema.org wants to set an end to dozens of independent vocabularies whose interconnections are undocumented and therefore not useful for publishers. It brings together several independently defined vocabularies, such as *LRMI*, *rNews* and *Good Relations*. The team behind schema.org states that the project can be considered as an approach to *Linked Data*.[16] Primarily, the vocabulary is used to annotate existing Web content. This requires the created vocabulary to often be flatter and less normalized than a purely database-oriented approach might be.

3.4 Information Retrieval APIs

People, locations and organizations represent a big part of important information that is placed on the web. Therefore semantic data should be included in order for search engines to find them more-easily and improve search results. Information Retrieval APIs crawl through content to find out which topics are mentioned in pieces of texts and return that relevant data. *Entity Extraction* or *Named-entity Recognition* are more specific terms for the described process and represent a subtask of Information Retrieval. Various APIs offer tools to extract distinct information and classify texts into pre-defined categories.

[16] http://linkeddata.org/

3.4.1 Alchemy API

The API *Alchemy*[17] follows the purpose of extracting semantic units from texts by making use of different algorithms and training data. It provides an output of different analysis tools such as *Keyword Extraction*, *Sentiment Analysis*, *Concept Tagging* and many more. Their entity extraction identifies cities, geographic features, companies and other typed entities from HTML documents or web-based content. It is mostly used in natural language processing (NLP) techniques to enrich a website's content semantically. The API claims to be unique in it's way of identifying the entities based on the combination of multilingual support, linked data, context-sensitive entity disambiguation, comprehensive type support and quotations extraction.

Alchemy API offers free use of their services for non-profit purposes and research. An API key is available after registration and allows a default amount of 1.000 extraction operations per day. The returned values from the entity extraction tool find around 293 different entity types. These represent only 45% of the existing schema.org entities and don't comply with their terminologies and definitions.

3.4.2 AYLIEN API

AYLIEN Intelligence[18] launched their first API in February 2014. They have a small number of nearly 200 users spread all over the world and the number is growing steadily. Their Text Analysis API consists of eight distinct Information Retrieval (IR), Machine Learning (ML) and Natural Language Processing (NLP) APIs that can be adapted to their user's needs. The Text Analysis API helps the developer to extract meaning and gain insight from content within documents. *AYLIEN* provides tools for the analysis of content in different areas such as Classification, Sentiment Analysis, Entity Extraction, Concept Extraction and many more. Developers can get started with the API and call it up to 1.000 times per day for free. The company plans to release new SDKs for other languages, besides Node.js, Python, Ruby and PHP, in 2015. Their Node.js API was published at the turn of the year from 2014 to 2015.

Following a knowledge-based approach for their Concept Extraction, AYLIEN has an amount of 128.935 entities available, which do contain a schema.org type representation. According to internal calculations the total number of entities that AYLIEN detects is much higher (around two million) but only a subset of them have been mapped to schema.org types. So concept extraction might return up to two million distinct entities, but only about 130.000 of them will have distinct schema.org types. Some are de-

[17] http://www.alchemyapi.com/
[18] http://aylien.com/

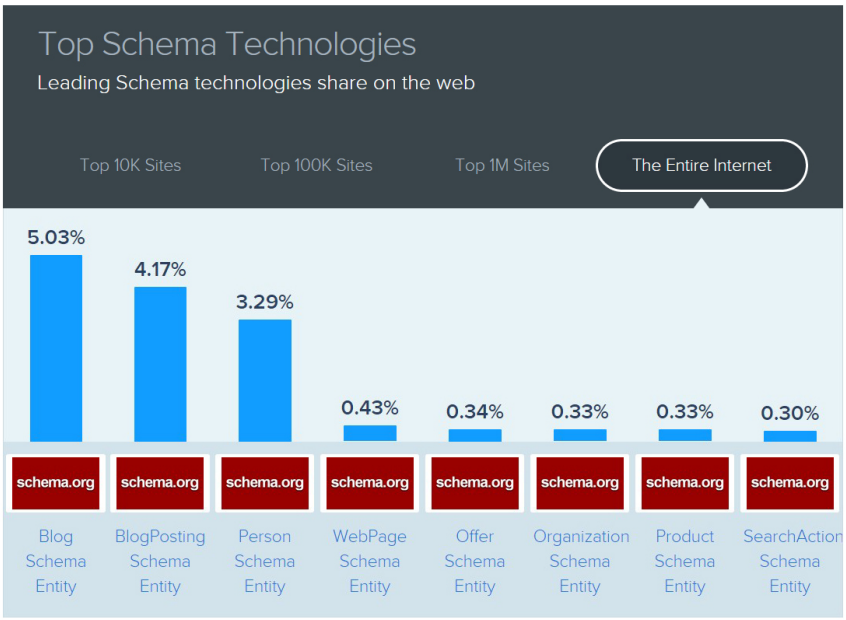


Figure 3.4: Leading Schema technologies share on the web [26].

scribed via DBpedia types (in the dbpedia.org/ontology namespace[19]) and for all other entities one would need to rely on what is available on Wikipedia (e.g. categories). *AYLIEN* has 42 distinct schema.org types as of now:

Person, Creative Work, Organization, Music Group, Music Recording, Place, Airport, Landmarks Or Historical Buildings, Body Of Water, River Body Of Water, Administrative Area, Music Album, Event, College Or University, Educational Organization, School, Product, Language, Mountain, Hospital, Lake Body Of Water, Sports Team, Web Page, Television Station, City, TV Episode, Movie, Restaurant, Book, Park, Festival, Radio Station, Stadium Or Arena, Sports Event, Museum, Government Organization, Shopping Center, Country, Library, Ski Resort, Hotel, Canal.

The team behind AYLIEN does not primarily focus on mapping specifically schema.org's vocabulary to their data-set collection. Therefore, the remaining entities are not included in their system, as of now. The current version of *AYLIEN's* Concept Extraction returns any entity that has a Wikipedia page. These may or may not have been mapped to schema.org's semantic vocabulary. But still, the shown types are assigned by AYLIEN via conservative mappings between DBpedia, Freebase types and schema.org types, which result in very precise but low recall values.

Figure 3.4 displays the top used Schema entities in the entire web. Comparing those types with the ones that AYLIEN covers, Schemas such as

[19] http://dbpedia.org/ontology

Figure 3.5: AYLIEN Concept Extraction API. Example [27].

BlogPosting, SearchAction and Offer, which place top positions in the ranking, are missing in this list and therefore not covered by the system. Although AYLIEN appears to have a very narrow system, they are still far from perfect. Looking at an example of a concept extraction from *AYLIEN* in Figure 3.5 we know as humans that the example sentence: "Avatar is an American television series that aired for three seasons on Nickelodeon from 2005 to 2008." talks about a television series, but the result in the concept extraction shows *Movie and CreativeWork* as main types. Word-sense disambiguation is a long-time existing problem which still causes systems to incorrectly identify a word's true meaning.

Chapter 4

New Approach

Various systems for supervised semantic analysis cover already a wide range of topics and offer tools for a fast and easy analysis work-flow, but their predictions are based on a manually pre-trained dataset and classifiers. These systems are expected to predict any given document's category from then on, which makes them less adaptable for contemporary, modern and advanced vocabularies. Furthermore, the biggest factor affecting the quality of the implemented predictions-tool and classifiers, is the quality of the training data set. This dependence is the motivation to create an automatic and resilient system for semantic text analysis. To avoid the known problem and challenge of WSD in fully automatic systems a hierarchical comparison of inherited hypernyms and entity relationships is planned to be included.

4.1 Concept and Architecture

In this chapter the own approach to create a fully automatic analysis tool for hierarchical vocabulary structures battling the known problem of word-sense disambiguation is described, as well as the design and theoretical background of the created system. The design is kept very simple and straightforward, as the true complexity lies in the preprocessing and analysis of the data itself.

4.1.1 Goal

Its artificial intelligence and independent behaviour is the overall goal of the system's architecture. All entities from the used vocabulary are detected and semantically included in a fully automatic manner. The main idea here is to have any sort of input text to take as a base for the entity analysis. Via Information Retrieval the most important data and valuable terms are detected and with the use of appropriate algorithms the best suited entity is identified for the specific target word or term. The form of inclusion via Microdata, JSON-LD or RDFa is left to be chosen by the user, while the

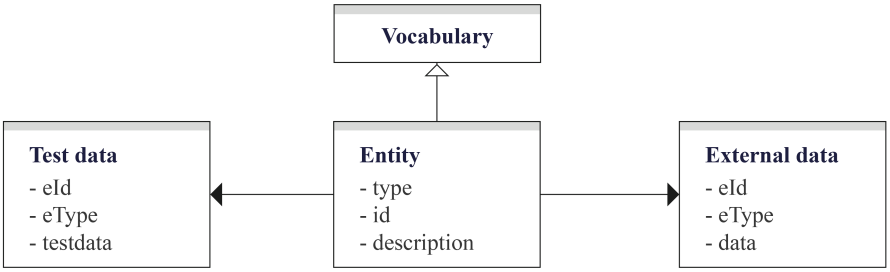


Figure 4.1: Basic system design.

main focus within the development process is set on improving the precision value of 69% (result of SemEval 2007 as shown in Section 3.2.1) in full automatic word-sense disambiguation systems to detect the true meaning of a term in focus.

4.1.2 System Design

Evolving around the given vocabulary, the system-design contains all entity types, as well as their hierarchical structure (Figure 4.1). Each entity should have information about its unique identifiers and be connected to external data. That data is retrieved from dictionaries (online and offline), wikis and search engine APIs (Section 4.3). Additionally, data for testing is created and paired up with each entity to evaluate the results of the completed preprocessing and use of chosen algorithms. Entity descriptions consist of general information, which are already given by the vocabulary itself, while the external data represents an extension of that data. In the following paragraphs the theoretical structure of the system and data collection is explained in more detail for better understanding.

4.2 Information Retrieval

To load the content and extract the main and most valuable information out of it, is the first step in the process of creating an intelligent system for semantic analysis. This step is done for the uploaded content which is to be analysed, as well as for the collection of entity identifiers. First up is the source document. If it includes HTML syntax, which will be the case in most uses for this project, all tags and syntactical data is removed to be left with plain text. With this raw data the words themselves are analysed in more detail without distracting the algorithms with special characters and code snippets or non-lingual expressions. Depending on the language a text might consist of multiple nouns, verbs and adverbs, numbers, dates or other numerical information and names of people and places. Alongside

these more or less meaningful words, also data without any special meaning to it, can be found within a text. As a first step "stop words" are removed to reduce the total amount of words to the most important and valuable ones.

For both data sets, which are the source document's valuable terms and all entity types of the semantic vocabulary, identifiers (descriptive information) need to be extracted from external sources. These are described in the following sub sections.

4.2.1 Dictionaries and Inherited Hypernyms

Semantically rich dictionaries, such as WolframAlpha[1] and WordNet[2], offer API's to retrieve information on specific queries for words and terms, as well as additional hierarchical information to their queried results. Each entity has a specific name that explains it in the most generic way, but still holds in some sort of keywords, which can be used to query more information out of them. The most important information that these dictionaries are offering is the option to extract inherited hypernyms. Those superordinate terms represent overall parent concepts in more abstract levels. The term "sea" in WordNet as an example returns the lexical definition of it as well as the inherited hypernyms, namely: "body of water", "water" and "thing". This concept is very similar to the hierarchical structure of described entities in semantic vocabularies. The correctly identified entity type "SeaBodyOfWater" in schema.org inherits properties from its parents "BodyOfWater", "Landform", "Place" and "Thing". This similarity of inherited hypernyms in both structures is used as main reference for similarity scoring and identifying the correct entity for a target term.

To ensure the best results for semantically correct types the data collection for the group of entity types needs to be analysed more carefully. Schema.org's entity "LakeBodyOfWater" includes two terms, namely "Lake" and "Body of Water", and inherits from "Thing > Place > Landform > BodyOfWater". Querying the term "lake body of water" does not resolve in any dictionary results. Therefore, the most unique data out of an entity name (in the given vocabulary of schema.org) needs to be determined. An obvious choice in the given example is to gather information on the term "Lake" instead of the entire entity name "LakeBodyOfWater". The hierarchical structure of schema.org's given semantic vocabulary offers a big advantage within this process. So, the descriptive name of each entity needs to be compared to its parents to extract the best suited identifiers. This step is also used to avoid having redundant information on sister entities, that might include the same data as their parents already have. To avoid querying the descriptive entity names word by word and therefore also analysing stop words, such as "of" in "Body of Water", an extra keyword and named

[1] http://www.wolframalpha.com/
[2] https://wordnet.princeton.edu/

entity recognition is used at this point. This offers the possibility to keep meaningful terms together and does not separate them as they might have a completely different meaning afterwards. Although the data and information collection in online dictionaries are already big, they are not perfected for terms containing multiple words. Retrieving information for the entity query "Music Video Object" does not offer any results. As there exist separate entity types for "Video Object" and "Audio Object" it is necessary to keep the strings together in order not to loose their meaning. One way to achieve this is to compare the divided entity names with the other ones of the vocabulary to check for synonymous duplicates. This might solve the problem of retrieving the same data and information for different entity types but not the problem of empty query results without any retrieved data.

Natural Language Processing tasks require often external sources of lexical semantic knowledge such as WolframAlpha or WordNet [5, p. 8]. In a time consuming and expensive manner these resources have been built up manually by experts. Collaboratively created data sources emerged with the Web 2.0 technologies and enabled user communities to build up these new kinds of resources. Traditionally, they are defined as Wikis.

4.2.2 Wikis

Online wikis, such as Wikipedia and Wiktionary[3], are an alternative to expert-based dictionaries with their by humans collaboratively created content. Therefore they often contain information and descriptions of data which are not found in general word by word data collections. This creates an additional source of descriptive data to be used to collect identifiers for target terms and entity types. They also offer more information in their query results which can be useful or fatal, as the results most certainly contain the correct meaning of a word. Wikis represent emerging lexical semantic resources, which can be used as substitutes for expert-made resources in AI applications. Their constantly evolving intricacy of interlinked articles represents a giant multilingual database of semantic relations and concepts, as well as a resource for NLP areas. The most useful and compact information is held within the abstract paragraph of the result pages. The abstract represents a compressed source of information that is held within the whole lexical knowledge page. Using this first paragraph of a Wikipedia article, as opposed to the whole article and overwhelming data on the page, is very likely to result in better precision values. Similar to search engines, they present multiple search results for multiple word meanings. This creates the risk factor of retrieving unnecessary information without any relations to the term in focus. To avoid the collection of multiple definitions, only queries resulting in single result pages are used for further analysis.

[3]https://en.wiktionary.org/wiki/

4.2.3 Search Engine Information Retrieval

But sometimes also wikis don't offer the needed information for specific terms, especially longer word combinations. To query the extracted data from the document in focus that might not find reasonable descriptions and definitions in online dictionaries or wikis, these terms do need to be compared to a larger document corpus. Search engines are nowadays close to perfect when it comes to smart and straight-to-the-point search results. The top five to top ten of all search results are usually the most suited and informative ones. As web pages, such as Wikipedia and news platforms, often already use up the first couple of result-slots, it is decided to skip the first two search results and still be able to get the most valuable information out of a query.

Information Retrieval via search engines is mostly designed for humans and possibly restricted by the developers to query multiple terms within a couple of seconds. A fully automatic system that uses IR via search engines requires to do multiple requests within a short amount of time. *Google's Search Engine API* restricted their API calls to a number of only 100 queries per day, which makes it, despite of its popularity and results, not useful for this approach. Fortunately other online search engines offer APIs to call for around 5000 times a month, such as *Bing*. The information within the title and short description of the top five Bing search results are collected for further analysis. As the system already included Wikipedia abstracts for the same query, the search results including Wiki information as well as news articles are excluded.

4.2.4 Entity Type Sub Pages

On the sub-page of a schema.org entity the inheritance of that type can be found as well as a short commentary description, which is initially made for humans to understand, what the entity type is all about. Further down a list of properties or attributes is given. These attributes can represent things and information that are related with this entity and placed in the context around the target word. The expected types of these properties are sometimes basic data types, such as Strings, Numbers or Dates, or other entity types, such as Media Objects, People or Organizations. Each given property has an own description, in form of a couple of sentences, again for humans to get a better understanding for how to use them. Therefore, the semantic vocabulary itself offers also a variety of information (descriptions, properties and hierarchical data, including inherited hypernyms), which is collected for identifier data-sets to identify the best suited single entity for a given target word or term.

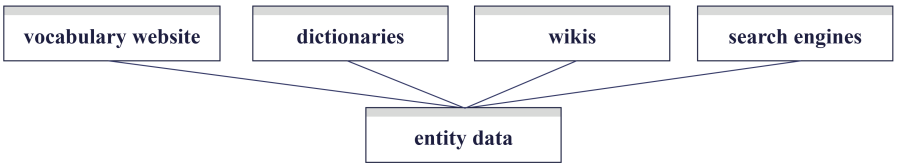


Figure 4.2: Information Retrieval for entity type identifiers.

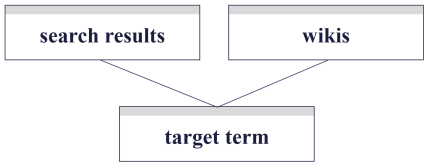


Figure 4.3: Information Retrieval for target terms.

4.3 Single Entity Detection

After filtering out the most valuable and meaningful terms in a document, the best suited entity types within a given vocabulary need to be calculated for them. To achieve the distinction between the various entity types each one needs a collection of identification objects (Figure 4.2). These can be terms or any unique identifier and are retrieved in a fully automatic manner, to follow the main goal of an independent system.

Most queries contain only a couple of words and most likely consist of proper nouns. That is why they are going to need more descriptive data as well. The analysis and Information Retrieval for these specific terms follow the same path as the entity type resource process from Section 4.3: collecting data for a query from dictionaries, wikis and search engine results. But the process of repeated API calls, ranking and filtering for each target word or term takes its toll on speed and performance of the automatic system. Knowing that the target terms consist allegedly of proper nouns and multiple words, the choice for resource options diminishes to wikis and search results (Figure 4.3). Online dictionaries retrieve supposedly not a lot of data for proper names of people, locations and organizations. Therefore they are not used as a form of Information Retrieval for this specific task. Wikis and search engine results on the other hand have a high probability of returning results with a high value. These results represent a base for the detection of identifiers and their synonyms before they are stemmed and compared to each entity type vector.

In the previous sections the data retrieval from dictionaries, wikis, search engines and vocabulary sub pages were described. The results are now to

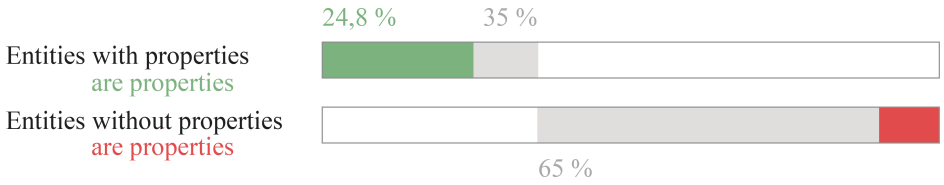


Figure 4.4: Entity-Property representation.

be analysed. An obvious choice, and the first approach to identify and differentiate entities at this point, is to locate properties within the context around the target words. These would simplify the process of finding the most appropriate entity with eliminating word-sense disambiguation. But in fact it is not as simple as it sounds.

4.3.1 Differentiation by Properties

Not offering specific attributes for each entity causes some problems. In Figure 4.4 it is shown that only around 35% of all entities within the schema.org vocabulary have property representations, from which approximately two thirds serve as properties themselves. Therefore 65% do not have any properties that describe them more precisely. And one tenth of all entities are included as attributes for other entity types without having any attributes. Another factor, that works against this approach of differentiation, is that some entity types represent properties within their own attributes. This would create an endless loop of trying to identify an entity, for a number of 15 entities within the schema.org vocabulary. Furthermore, the missing attributes and small descriptions do not represent enough data to differentiate each entity from another. The fact that not all entity types have their own properties to identify them properly makes the extraction of unique identifiers more difficult. But also if all entities would have unique descriptive properties, they might not be found within the context around the target word.

4.3.2 Differentiation by Context

A word usually gains its meaning through the context and words that surround it. Unfortunately that does not apply to all cases. Some queries or strings might include target words without actually referring to them, for example with list items. These end up with no meaningful word relations to extract. A news article could talk about a theft in a known supermarket of some small town. But the whole article might refer to the town and the burglar himself. Therefore, the supermarket's well known name could be mentioned just once, and no entity properties would be found around

that target word to describe and identify it properly. But the terms and nouns representing examples are often proper words that do not need the content around them to identify their meaning. The name is probably already quite unique and substantial itself, such as companies (Google) or celebrities (Barack Obama). They probably will not be found in dictionaries, but in collaboratively constructed knowledge sources, such as wikis, or on informative and educational websites.

The analysis of different word types and their results leads to the division of queries into four main word groups: “proper nouns”, “nouns”, “verbs” and “alternative values”. Depending on their queried results from dictionaries, wikis and search engine requests the type is identified. Proper nouns are much likely not to retrieve any results from dictionaries, but from wikis and search engine results. The inverse scenario is found for verbs. Alternative values are much likely to have only search engine results and nouns get their most valuable results from dictionaries.

If a term then still does not retrieve a lot of valuable identifying data, nouns located close to the target term (window nouns) are analysed. This eliminates the risk of not retrieving any data for the upcoming identifier selection for the document in focus and all entity types of the used vocabulary.

4.3.3 Term Frequency and Inverse Document Frequency

After collecting enough data from various resources the best suited identifiers need to be selected. By retrieving information with rather specific queries per entity type a big step towards good differentiation for word-sense disambiguation is already taken. To collect and determine unique identifiers for each entity is for a fully automatic system the most challenging part. The data retrieval and selection stays an unsupervised process to follow the goal of full automatism.

In Information Retrieval the most obvious choice for best practice IR is to include the statistics for Term Frequency and Inverse Document Frequency (short TF-IDF) to determine the terms with the most valuable information in a document compared to a corpus. The goal in this process is to identify the mutual information between a text (the collected identifiers) and computed keywords that represent a specific content without loosing any important information. TF-IDF is therefore used as a channel to maximize the value of mutual information between the described elements.

Term Frequency and weighting

Different variation schemes for weighting the result of the TF calculation can be chosen to determine its exact value. In this project the *logarithmically scaled frequency* is used to avoid favouritism or uneven results between long and short documents.

Inverse Document Frequency

As not all words are equally important a factor has to be included that reduces the TF weight for a term, depending on its collection frequency. Therefore, the Inverse Document Frequency (IDF) of a rare term is high and the one from a frequent term is likely to be low. By combining the definitions of TF and IDF a composite weight for each term in every document is produced with the TF-IDF weighting scheme.

TF-IDF

Using the formula for Term Frequency and Inverse Document Frequency from the Equation 2.2 the calculated values for TF and IDF are multiplied to determine the most valuable terms within the identifier collections. At this point it is possible to see each document as a vector with one component corresponding to each term in the dictionary, and a calculated weight for its importance or relevance. For terms that do not occur in an identifier collection the weight is set to zero. The resulting vector form will prove to be important for scoring and ranking at a later point.

One occurring problem with the calculation of TF-IDF is the term-handling as bag of words. Not only documents and single terms need to be taken into account to define the appropriate event spaces for the required probability distributions, but also queries and terms [18, p. 7]. Many researchers are eager to replace the heuristic in the IDF component with some reasonably-constituted theoretical argument, in order to somehow explain why it is that the probabilistic function in IDF works so well. But other than being a small mystery, the TF-IDF formula is still used in most search engines as well as in this project as a ranking method.

4.3.4 Single Entity Identifiers

With the calculated TF-IDF the retrieved data within each entity description is compared to each other. The more specific terms and words for each entity type are selected as the most suitable identifiers. Then they are used for data comparison in vector space models (VSM[4] further information in Section 4.4).

Synonyms and Stemmers

Each term takes part in a vector representation of an entity. At this point it is crucial to the forthcoming data comparison that the match for the correct entity type achieves the highest confidence score for mutual information. As terms have different forms of appearance (grammatical alterations and word variations), it is decided to include not only synonyms of words but also

[4]More information on TF-IDF and VSM: http://nlp.stanford.edu/IR-book

to use a so called *stemmer*.[5] Stemmers reduce words to their most generic stem. A rule system removes parts of words to a compressed version of a word group with as little syllables as possible. The new Snowball stemmer [33] is a derived version of the original Porter stemmer [34] from 1979 and recommended for usage due to its improvements. This step is necessary to collect as much information with having the smallest amount of words as possible. The same stemming-procedure is repeated at a later point when the resources of search queries are analysed and compared to the identifiers of single entity types. After assembling and filtering out all data resources for each entity the results are saved in a database following descriptive models for the upcoming data comparison with target words and terms. A detailed description of the database saving process can be found in Chapter 5.

4.4 Data Comparison and Confidence Scoring

The Vector Space Model (VSM) finds much use in scoring and confidence measuring for the comparison of the document term data-sets and the entity identifier collections. These documents and lists are represented as vectors in a common vector space. A small document could have a vector representation as follows: $[the, grass, is, green]$. The vector of all terms occurring in the corpus D is denoted

$$T = \text{sort}(d_0 \cup d_1 \cup \ldots t_j \cup), \tag{4.1}$$

for $j = 0, \ldots, |D| - 1$. This vector represents a term-collection from all documents within the corpus. A vector of terms contained in a specific document d is denoted by $\boldsymbol{v}_d$, with

$$\boldsymbol{v}_d(i) = \begin{cases} 1 & \text{if } t_i \in d, \\ 0 & \text{otherwise,} \end{cases} \tag{4.2}$$

for $i = 0, \ldots, N - 1$. Thus $\boldsymbol{v}_d \in \{0, 1\}^N$ and $\|\boldsymbol{v}_d\| > 0$. Then the set of documents in a collection is viewed as a set of vectors in a vector space. Each search term represents one axis. A search query will have the highest similarity score with the entity term axis that includes the highest number in equal identifier terms. The similarity between two documents and therefore two vectors is defined by how similar their pointing in direction is (Figure 4.5). This is done by computing the cosine similarity of the vector representations as follows:

$$\text{sim}(a, b) = \frac{\boldsymbol{v}(a) \cdot \boldsymbol{v}(b)}{\|\boldsymbol{v}(a)\| \cdot \|\boldsymbol{v}(b)\|}. \tag{4.3}$$

[5] Stemmer: the form of a word that remains after removal of all inflectional affixes [28].

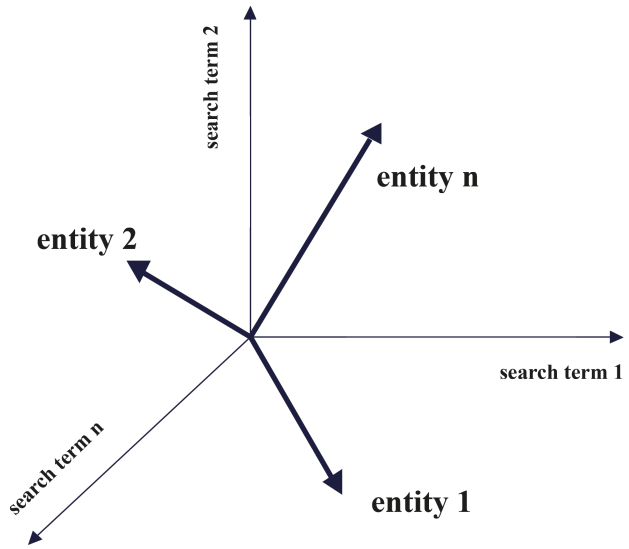


Figure 4.5: VSM example for cosine similarity.

That calculation avoids the bias caused by different document lengths. To measure the similarity of two data-set collections (e.g. a specific entity and a target term), their inner product (sum of the pairwise multiplied elements) is divided by the product of their vector lengths. The division has the effect of normalizing the vectors to unit length and only the angle, or more precisely the cosine of that angle between the vectors, accounts for their affinity. As documents always contain at least one term, a division-by-zero is not possible. Documents that do not have a single word match get assigned a similarity value of 0. This results out of the orthogonality of their vectors. Therefore documents with a similar vocabulary get higher values, up to 1 if the documents are identical. Finally, the resulting score between a query and the various documents within a corpus are used to rank the results.

4.4.1 Group Entity Identifiers

At this point the collected entity detection data from Section 4.3 is to be compared to the target word or term in focus. The comparison of the results from the data resource collection with the single entity identifiers puts high pressure on the correctness of having selected the correct terms for each entity type. Word-sense disambiguation is still a risk, as a word's meaning might change in different context settings. To avoid the determination of wrong entity types the text around a target word needs to be included into the analysis as well. But the context does not always include enough information on the specific entity to differentiate it from its sister types. However, a text is often written with a concept and these overall concepts

can reflect the hierarchical structure of a semantic vocabulary. Every entity starts with the abstract concept of being a *Thing* and branches down into more specific groups and types. Therefore, it is desirable to narrow down the possibilities of the correct entity type for a target word once you know the general notion behind a text. So, instead of comparing the target words resource data to single entity identifiers, it is chosen to be more reasonable to compare it first to their highest parent types and then work down the path to more specific branches in the hierarchy tree. This eliminates not only the chance for word-sense disambiguation but also reduces processing time to not having to compare each and every entity data in every loop for target word comparison.

Each group needs a small set of identifiers which determines the overall concepts as a result for different target words. Therefore, the already assembled information on a parent entity as well as all of its children is put together. The created data set is now ready to be analysed to reduce its size of terms and words to the most specific and descriptive ones for the overall concept in focus. Terms that appear repeatedly within the set are placed next to proper names on top of the list of important identifiers. All remaining words are compared to each other and to the sibling sets of the same-level concept-groups. This process is imitated in each hierarchy depth to collect the most significant terms for every concept with proportionate detail. The resulting data sets are then used as a comparison base for a specific target word.

4.4.2 Inheritance Similarity

The same procedure is done for a target word's collected hierarchical structure of WordNet's inherited hypernyms. Similarities between a target term's dictionary hierarchy and an entity's inheritance structure is weighted with a higher value than the simple identifier list between the data-set collections. The same word roots are seen as important selectors to eliminate word-sense disambiguation. Therefore, a high resemblance is assumed between similar inheritance structures of two terms.

The determination of the most suitable entity type does not start with comparing the data to a single entity type, but to the overall parent group. This identification will resolve in the elimination of word-sense disambiguation and fasten up the test routine itself. The already collected information on the first depth in hierarchy is used at this point to compare the most valuable identifiers of all entities within one parent group with the recently calculated ones from the search query. In the vector space of all parent groups the target word representation vector is now compared to the other vectors in the space. After selecting the closest one with the best similarity value the parent will open up a new vector space with only the entity types that are inherited from it (Figure 4.6). To narrow down the number of

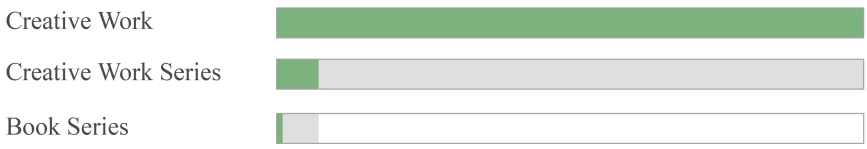


Figure 4.6: Group identifier loop.

possible entities for a specific search term the process is repeated until the lowest level of parent groups possible. The schema.org vocabulary hierarchy has a maximum depth of four levels, which keeps the number of loops per test on a lower value. Once the lowest parent group is determined the vector space will change from parent group vectors to single entity vectors. At this point the error factor is held quite low as the correct category is already determined. This of course depends on the quality of chosen and calculated identifiers for each single entity, group and target word representation.

At the end of one target word loop the system is left with an entity type that represents the highest confidence and vector space score. This one is returned to be then included into the original document. With the incorporation of all found entity types the document is semantically enhanced and represents a data collection that is readable and ready for interpretation by machines.

Chapter 5

Implementation

After defining schema.org as the main vocabulary setting the main goal to have an unsupervised program and researching through the theoretical background of creating such a fully automatic system some modifications are necessary. To have a vocabulary with pre-defined entities for classification does not fall into the category of an unsupervised system. However, the remaining parts of the program stay without supervision. Therefore the adapted goal, following correct terminology is now set to semi-supervision. The subsequent goals of the developed system are its independence and intelligent behaviour. The included components were chosen with the focus of having it act with as little human input as possible to ascertain the possibilities and restrictions of a fully automatic semantic analysis program. The JavaScript based code counts different important components to its core which are described in the following sections.

5.1 Framework

Traditional scripting languages, such as PHP, are widely used for web applications and are based on requests. But the server-side JavaScript technology in NodeJS[1] gained more popularity and interest in the web developer community. The requirements to the framework for the developed system aren't strict or demand high complexity. Various semantic libraries, tool sets and support programs are provided by a majority of frameworks. Wolfram Alpha, WordNet and Aylien are the main APIs and tools that represent a necessity for inclusion. NodeJS fulfils the decisive requirements and is therefore respectfully chosen to be the framework in use.

[1] https://nodejs.org/en/

5.1.1 NodeJS

NodeJS is an open source and cross-platform runtime environment for server-side and networking applications. The provided event-driven architecture and the non-blocking I/O API optimizes an application's throughput and scalability. The JavaScript environment is often used in real-time web applications and represents therefore a good base for the development of the client-side system handling. *Express*[2] is a NodeJS web application framework with a quick and minimalist background. The flexible framework provides a robust set of features for different kinds of web applications and is a good choice to create the user-interface for the system. *Express* is internally based on *Connect* and extends it with numerous missing aspects. These include REST web services for routing and also various configuration features. For development purposes command line interaction with the system is provided to update, pre-process and test the data and code. It represents the main interface at this point, to ensure the focus on advancement in code development. Therefore the express environment is taken out of the system but will be included again at a later point in time.

5.1.2 Node Modules

NPM[3] (Node Package Manager) is the pre-installed package manager for the NodeJS server platform. With it modules and programs from the NPM registry are installed and used. By organizing the management of third-party NodeJS programs and organizing the installation process, it represents a help for developers to build up their code in a fast and easy way. The used packages in the developed system can be divided into four categories:

- database,
- text analysis,
- data collection and
- helper packages.

All node modules can of course be seen as general helper packages as they facilitate processes, such as reading files and retrieving data via jQuery or fasten up the implementation development by including already optimized algorithms and code snippets. Some of the explained data retrieval, analysis and filtering is done with the installed text analysis and data collection modules. The exact use of all included packages will be explained in the following sections while describing the system itself.

[2] http://expressjs.com/
[3] http://www.npmjs.com/

5.2 Database

To build the data foundation of the system the vocabulary's structural and hierarchical information is collected. Due to its popularity and maintenance by the biggest search engines in the world, schema.org is chosen as the vocabulary for the semantic analysis of web pages. Schema.org's homepage includes all necessary information to describe the type structure and its hierarchy so that all entities can be identified and saved in a fully automatic manner. In the early stages of development all information was stored in simple JSON-files, as the complexity of the data-structure is small and straight forward. After refactoring and considering changes to clean up and simplify the system's design the data was transferred into a database. Options for complex relations and table connections are not vital for the saved data. Therefore, a simple and fast connecting database platform is sufficient.

5.2.1 MongoDB

All entity types and further semantic information are saved in tables within a *MongoDB*[4] database. With its cross-platform and document-oriented structure it is classified as a NoSQL database. JSON-like documents, so called *BSON*[5], and dynamic schemas enable a fast and easy integration of all high-value data. The table structure is held very simple, having only small relations via IDs between them. For further facilitation *Mongoose*[6] is added to the database component. It provides a straight-forward schema-based solution to model the application data and include built-in type casting, validation, query building, business logic hooks and more. Both, MongoDB and Mongoose, are installed respectively via their provided node packages. Mongoose offers a model system in which the structure for each table is saved as BSON.

```
1 var mongoose = require('mongoose');
2 var db = require('../dbConnection');
3
4 var Vocabulary = mongoose.model('Vocabulary', {
5     type: String,
6     depth: String,
7     comment: String,
8     properties: [String]
9 });
10
11 module.exports = Vocabulary;
```

After requiring the necessary packages and connecting to the database (L1 and L2) the collection structure is set. Each model provides the tables with

[4] https://www.mongodb.org/
[5] http://bsonspec.org/
[6] http://mongoosejs.com/

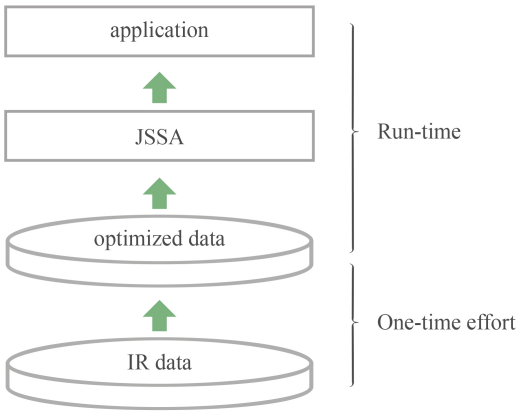


Figure 5.1: Application structure.

an automatic ID for each element and the possibility of storing different valued columns, such as Strings, Arrays and sub-JSON structures (L5 - L8). As the system requirements contain a fast fetch of tokenized data within arrays the option of saving and retrieving arrays as a data-type proved itself to be very useful. The collected and listed information for each entity type is stored in a quick and easy manner. Simply an array is needed without having to stringify it or break it up into its individual components.

5.3 Application Structure

Within the automatically created NodeJS application a folder structure containing all basic components is built. Node packages and modules are saved in a separate folder next to the general information file. This dataset saves all dependencies as well as the node engine and version numbers. Alongside of these parts that assure a robust app foundation the different components for data retrieval and analysis are placed inside the app folder. These components are divided into four categories: models, data, preprocessing and updates. The already described database models serve as the first part to set up the system in its structure. In Figure 5.1 the overall application structure is depicted. Via Information Retrieval data is fetched from dictionaries and other sources in an one-time effort. This base represents the database foundation from which the optimized data is created from. The actual run-time part of the system takes the newly optimized data as a base. Subsequently, the developed JavaScript-based Semantic Analysis (JSSA) tool is used to identify entity types for specific target words within the application.

5.3.1 Models and Data

An own model is created for each table and saved within a folder called *models*. This explicit denotation is used because the database fetches the table structure from this by the module specified location. The distinct models are explained further in the sections for Updates (Section 5.3.2) and Preprocessing (Section 5.3.3). Next to the models folder a collection of static data is placed. That specific data does not represent any necessity to be saved within the database. It contains specific information of the chosen vocabulary and is therefore not important for a general system of semantic analysis. The vocabulary of schema.org offers almost 700 entity types up to now. During the process of development over 20 new entities have been included into the hierarchy but none of them were part of the category *Data Types*. In the schema.org vocabulary the Data Types represent string, number, boolean and date values that can be included as properties or attributes of entity types. But they do not contain any big information value for the entity differentiation process (Section 4.3). As these entities need special treatment during the analysis and preprocessing phase, the list of these data types is saved separately.

Furthermore, a list of already optimized entity types is saved for the inclusion of the *AYLIEN Text Analysis API*. Its *Concept Extraction* tool enables the system to start with a well developed base for single entity detection (Section 4.3). Out of all Schema entities, which are in count over 600, the extraction tool is able to resolve 42 of them correctly with a high precision value. This builds the foundation of the entity recognition and classification. AYLIEN established a huge database of words and their meaning by collecting terms in texts all over the web. Each of the 42 mapped entities for schema.org[7] has over one thousand usage examples for determining their semantic meaning and excluding word-sense disambiguation. This high database maintenance forced them to provide resources for only a small number of schema.org entities. But the results for that little share in terms of precision makes it an obvious choice of use as a base for the semantic system. As the number and types of the covered entities aren't officially published in an open list, the collection has to be saved in a separate file to focus on the detection of the remaining entities and classify them automatically while eliminating the sense disambiguation problem in a fully automatic manner.

5.3.2 Updates

To keep the vocabulary and hierarchical structure data up to date, scripts for updates are implemented and made available to call at any point. The

[7]This number was given to the author after contacting the company AYLIEN via their offered user chat on their website and e-mail. The distinct supported entities are not available online. Therefore, no reference is made.

first data collection and update script represents the vocabulary and its hierarchy. Each entity is retrieved automatically from the overview sub-page of the schema.org website. The clean construction of the site allows the system to retrieve data for each entity type in form of a short commentary description and properties in a fast and easy way. In order to make the system work for automatic updates on different vocabularies their structural representation will need to be analysed separately. Another solution is the development of an automatic vocabulary-representation scan and a subsequent analysis to extract the "structure", "hierarchy" and "relationships" between the various entities. While fetching the website's data the hierarchy and relations between the different entity types are saved. This is done to enable the later comparison and group entity detection to identify the best suited entity for a specific target word. But the update-commands for the vocabulary and hierarchy need to be used with discretion. Some entities do not have a unique name in the sense of appearing multiple times within the hierarchy of the vocabulary.

In Figure 5.2 the occurrences of the entity type *Medical Audience* are shown as an example. Being four times present with the same type-name in one vocabulary makes the distinction by name quite difficult and eliminates it as an option to do so. Therefore, the IDs that are created automatically by updating the hierarchy and storing them into the database are very important for referencing the exact type of entity with their parents. All sub-information for the vocabulary's entities are based on and differentiated by this given ID to not create unnecessary problems for disambiguation. This of course creates a big dependency between the hierarchy and all other collections. When a new entity type is created by schema.org the vocabulary and it's hierarchy need to be updated. This creates changes in the automatically created IDs and is therefore avoided by just adding the new entities to the end of the vocabulary list. Before the data was stored within a database the vocabulary got stored in JSON files. The hierarchical storage of the items from top to bottom made the grouping of all entities by main entity types easy and fast. With the change from JSON files to database tables the grouping by chronological order was no longer a possibility. At this point the IDs emerged to a higher importance and were then used as the main piece to connect the relations between all entities and their hierarchical structure for the arrangements in groups.

But all other information resources have also updates in their data collection to describe certain words and terms. An overall update, including all data sources for information analysis and definition, is therefore recommended. A system update refreshes all data to stay up-to-date and provides the best results possible for the semantic analysis. The sub pages for each entity type on schema.org and getschema.org contain in several cases next to their descriptions and definition also examples for usage. These are fetched and saved to facilitate the testing of the system by retrieving test examples

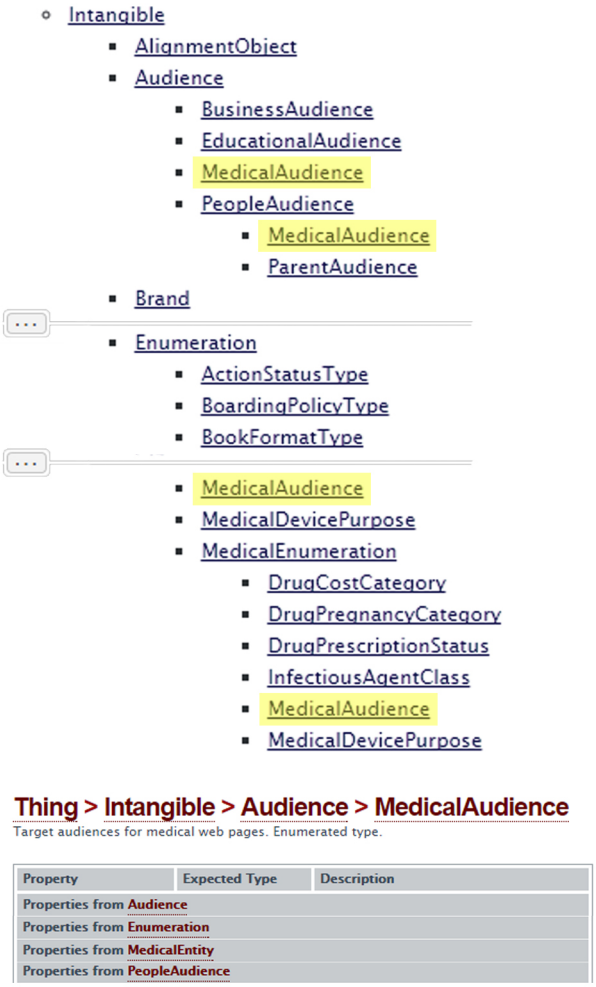


Figure 5.2: Schema.org Example: Medical Audience.

with correct results for most entity types. A detailed explanation on how to obtain the valuable data for testing purposes is described in Section 6.1.

5.3.3 Preprocessing

After updating the general structure of and within the vocabulary it is time to group and analyse the retrieved data for each entity type. As described earlier the way to gain more information, especially unique identifiers, for the various entities is to make use of quick access dictionaries, wikis and search engine results. Retrieving the data beforehand makes the process of comparison afterwards a lot easier and faster. WolframAlpha and WordNet offer online and offline APIs respectively with a generous restriction of calls

per month. The combined ability of both systems to retrieve a words definition and word relations via API requests is used to collect more information on the vocabulary's items. The general approach is to make use of the hierarchy behind the vocabulary of both systems. Similar to schema.org, the dictionary APIs are built on a hierarchical structure from which the inheritance and relations between words are extracted. The word "dog" as an example is part of the word group "animal", which is part of a word group itself: "living organism". This internal connection between terms is taken as an advantage to identify word groups and eliminate WSD in an early stage of semantic analysis. Of course, this does not apply to proper nouns for which search APIs are used. Proper names and terms, such as found in celebrity names or termed lakes, do not profit from these deep internal word connections. These labels occasionally have relations to general concepts such as *Person* or *Landform*, but almost no information on them exists in general public dictionaries. For rather basic terms, which appear in such word collections, enough data is retrieved for the semantic analysis of them. Therefore, the combined knowledge sources of classical wordnets (wisdom of linguistics) and collaboratively constructed knowledge sources (wisdowm of crowds), such as Wikipedia, is taken as a foundation for data requests. This allows a combined approach to evaluate semantic relatedness with a broader spectrum and choice of definitions, word relations and sense disambiguation.

Wolfram Alpha API

Through an existing node package the Wolfram Alpha API[8] is easily integrated and called with a simple request.

```
1 var result;
2 wolfram.query(entity, function (err, result) {
3     if (err) throw err;
4     if(result !== null) {
5         if(typeof result[0] !== "undefined") {
6             for(var i = 0; i < result.length; i++) {
7                 if (result[i].title == "Definitions" ||
8                     result[i].title == "Inflected forms" ||
9                     result[i].title == "Synonyms" ||
10                    result[i].title == "Narrower terms" ||
11                    result[i].title == "Broader terms"  ) {
12                      result = result.concat(removeStopWordsTokenizer((
      result[i].subpods[0].text).replace(/(\||verb|noun)/g, "")));
13                }
14            }
15        } else {
16            console.log('Empty result.');
17        }
18    } ...
19 });
```

[8] http://products.wolframalpha.com/api/

Next to the basic definitions of a word, other linguistic information can be queried. To widen the range of comparable data “inflected forms”, “synonyms”, “narrower terms” and “broader terms” are collected as well. These offer a broader analysis of words and relationships between word group branches. A simple and straightforward term comparison is often restricted through its lack in word-variety. The meaning of a term is put against stored classifying objects. These should project an entity’s identifiers to find similarities and equalities between them and the term in focus. To widen the spectrum for this resemblance analysis a bigger variety through synonyms and inflected forms is used. Not to overload the data collection and slow down the process of analysis it is necessary to select the most specific and eloquent terms. The TF-IDF algorithm is here again a good choice, although it needs to be slightly modified. A substantial variety throughout the data collection is needed, while still maintaining a compact set of terms. Therefore, the additionally collected data is analysed separately and subsequently brought together. The overall collected information is saved first in a model collection before being analysed. This allows an overall analysis for further grouped entity collections and supplementary term comparisons. The then analysed and in their size reduced entity data sets are saved in a separate collection for obtaining the needed information quickly. The same process for semantic analysis and filtering is done with the WordNet and search engine results data retrieval.

WordNet API

WordNet[9] is another classical linguistic data collection, which represents relatedness of words next to their definitions and disambiguations. NPM offers multiple models to call WordNet’s API. It is used with a downloaded vocabulary, to make it accessible for requests offline as well as online. Therefore, fast querying for various terms is made possible via the availability of the offline dictionary. Similar to WolframAlpha, WordNet is fetching definitions and domain types to retrieve a wide range of information on the query terms in focus.

```
var walk = new wn.Word(entity, "n");
walk.getSynsets(function (err, data) {
    var definitions = [];
    var domainTypes = [];
    for (var i = 0; i < data.length; i++) {
        var curr = data[i];
        definitions[definitions.length] = curr.definition;
        domainTypes[domainTypes.length] = curr.domain_type;
    }
    ...
});
```

[9]https://wordnet.princeton.edu/

Domain types represent the word group a term inherits from. This serves as a comparison base for word-relatedness between hierarchical groups of terms in semantic vocabularies. The expert-made resources in WordNet contain concepts that are connected by lexical semantic relations. It offers to show synsets and semantic relations as well as lexical relations for words and options for sense disambiguation, if a word has multiple meanings. The differences in word relation choices between the two described dictionaries permits a broader spectrum for word analysis and the selection of ideal identifiers for entity recognition. The documents of the Semantic Web are usually created by human beings, such as in Blogs, News Papers or Social Media Platforms. Therefore, they represent actually much more natural language documents than theory would make one believe. Collaboratively constructed resources, such as Wikipedia and Wiktionary, emerged themselves to be valuable semantic knowledge bases. They have a high potential in diverse Natural Language Processing tasks but differ in their access mechanisms to the knowledge stored in these semantic knowledge bases, as well as in the possibility to identify word relations.

Wikipedia

A representation of a valuable lexical semantic knowledge base, created collaboratively by non-professional volunteers on the web, is Wikipedia[10]. Due to its vast increasing size and significant coverage of past and current developments it is used as a semantic resource. Collaborative Knowledge Bases (CKB) are oppositely to Linguistic Knowledge Bases (LKB) less strict in their approach of construction. While LKBs are usually very costly to maintain and construct (except WordNet), CKBs are released under a license that grants free usage. They keep their content up-to-date, while the release cycles of LKBs do not reflect recent or more current events. The possible high benefit of using CKBs queried result for NLP comes nonetheless with a big challenge. Retrieving unstructured or semi-structured data with a high probability of noisy information creates a new challenge for itself. Furthermore, CKBs content relies on social control for the assurance of accuracy and comprehensiveness. The most valuable information for semantic analysis on a Wikipedia page is mostly found in the first paragraph. Representing the abstract of an article, it is basically a small summary of the page content and is easily fetched (L1 - L2).

```
1 var opt = {query: query, format: "json", summaryOnly: true};
2 wikiParser.searchArticle(opt, function(err, htmlWikiText){ ... }
```

The resulting reliability of its referential correctness to the target word in focus is a risk that needs to be taken. To search articles by keyword is an optimized structure that Wikipedia proposes to its users. The created

[10]https://www.wikipedia.org/

system iterates through all found articles and retrieves valuable information through the analysis of links, categories and redirects.

Search Engine API

Next to collaborative and linguistic knowledge bases also smaller web pages do present informative descriptions and definitions of various terms. Via user ratings and clicks on their pages to represent their popularity and usefulness they reach higher positions on search result pages. Longer phrases or rather contemporary word combinations have presumably a longer time-span to be added to public dictionaries or wikis. Blogs and user-driven platforms that are no CKBs include such new terms possibly faster and represent an additional source for IR.

Google is by far the most popular search engine in the world and has, according to statistica.com[11] a market share of 88.44% in April 2015. But through its popularity and power in the field the tools with and around it are often restricted for usage. Their search API calls are restrained to only 100 per month, while Bings API has a limit of 5000 transactions per month. That big difference made the choice to use Bing's search result module[12] very easy.

```
1 BingAPI.composite(options.hierarchy[options.c].humanizedType, {
2       top: 6,
3       skip: 2,
4       sources: "web"
5   }, function(error, res, body){
6       var searchRes = "";
7       for(var j = 0; j < body.d.results[0].Web.length; j++) {
8           searchRes = searchRes + " " +
9                       body.d.results[0].Web[j].Title + " " +
10                      body.d.results[0].Web[j].Description;
11      }
12      ...
13  }
14 );
```

The most valuable information is retrieved from the first couple of search results (L2), which contain the most valuable information in reference to the search query and target term. As the top result slots are often taken by either Wikipedia, news platforms or the web page in focus itself it is wise to exclude them from the analysis (L3). The request allows the option to ignore news articles right away as they presumably include the target term but not primarily valuable information on it (L4). The extracted title and description of each rich snippet is then used for analysis and therefore the extraction of relevant data.

[11] http://www.statista.com/statistics/216573/worldwide-market-share-of-search-engines/
[12] https://www.npmjs.com/package/node-bing-api

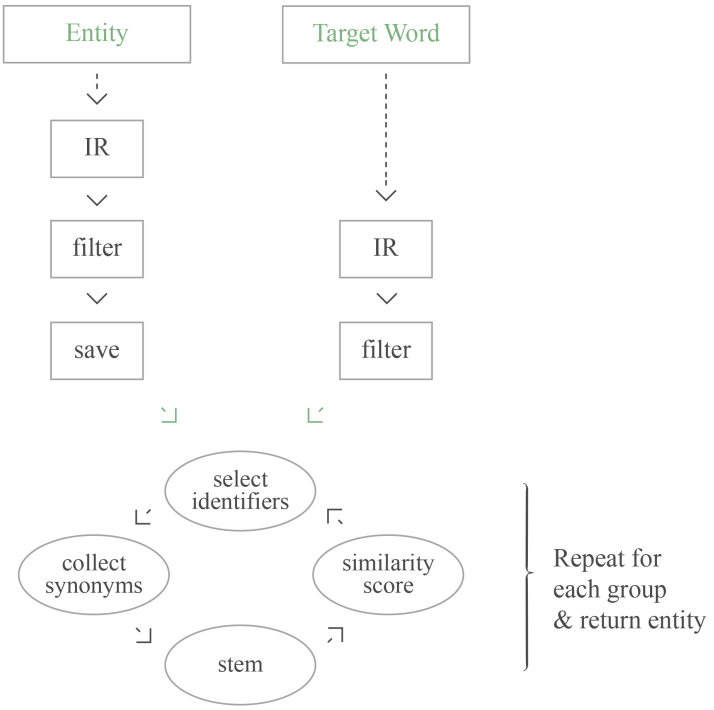


Figure 5.3: Entity and target word analysis.

5.4 Text Analysis

Information Retrieval via search engines, collaborative and linguistic knowledge bases leads to a substantial amount of data. The data-set is taken as a base for the upcoming semantic analysis to identify entity types for specific target words. All collected entity and target term identifiers go through a similar analysis process. Both data-sets collected their data from Wikipedia, WolframAlpha, WordNet, Bing and the vocabulary in use, which is schema.org. The final comparison is based on the highest similarity score and calculated via a term vector space model (Figure 5.3).

5.4.1 Filtering

Each target term collection is run through a stop-word removing *tokenizer*. The list of stop words is based on the standard version for the English language and modified by the addition of several terms. This is especially important as the Information Retrieval includes words and terms that represent the dictionary, vocabulary or information resource that the data is from. To fetch information from Wikipedia for example includes the word

Wikipedia itself and various brace-number combinations that are linked to resources within the article page. Numbers do also present a certain insignificance to the data set for entity types and their entity detection. Dates or numeration within content is seen by the TF-IDF algorithm as meaningful. Its rare usage over documents ranks it high up on the word-importance level. The identification for entities on the other hand is based on term weighting and comparison. Numbers and dates do contain a lot of meaning for various terms in their definition and description. But they do not serve as a valuable identifier for the systematic match up of an entity description.

The collected data for every entity is saved in a database collection beforehand. This is done for reference to group and compare as well as to calculate the similarity score. During run-time the data-set for each target term is retrieved and analysed to be then compared to the single or group entity data collection.

5.4.2 Single and Group Identifiers

After the removal of all insignificant words from the data set, the terms are listed by their term frequency. Although more frequent words do not automatically imply higher importance they do represent an indicator of relevance. With their repeated appearance in definitions and descriptions for a target word or term they develop a meaningful position and spot in the ranking for identifiers. Additionally, the inverse term frequency (IDF) is calculated to determine which words are the most meaningful to a specific topic, such as an entity type. This suggests that also rare terms do represent significance. All words in the entity-data collection are compared to each other to determine the most valuable ones. Similar to this, the IDF is used on the target term data-set. In difference to the entity data collection the comparison-set is based only on one document (the collected data).

The now rather big word collection for all entity types is now analysed repeatedly through a loop. To start the identification process the vocabulary hierarchy is broken down into its main branches. These represent all overall categories from which the target term should find its most fitting parent type and concept. Having a book as an example for a target word should result in a high similarity score for the overall category of “Creative Work”. Every branch assembles their child-entity data sets to create a new temporary collection. These go through a loop to calculate a similarity score for the best fitting entity type. In each repetition the group data-set gets more specific, depending on the parent concept with the highest similarity score in the round before. At the beginning of each loop the top terms of every category are selected and marked as most valuable. Subsequently all duplicates are removed to reduce the data-set of each group in its size. The same process is repeated for the target word data collection to prepare it for the upcoming comparison.

5.4.3 Synonyms and Stemming

Each data-set consists now of the most valuable terms to represent every category. A wider spectrum and better result in word comparison is achieved by the inclusion of synonyms and similar terms for each data-set component. This allows the possibility of a bigger variety in words with the same meaning and distinct representations of it. But at this point the collections are still likely not to result in high similarity scores. Identical words, such as *tree* and *trees*, are still seen as different although their word stem is the same. The implemented Porter Stemmer reduces the data collection of each category rather significantly and increases the score numbers for similarity.

Single and group entity identifiers are now converted into the same structure and with the same base as the target term data collection. The term vectors for each entity group and target word are now compared within a space model to identify the most similar data-sets. After the determination of the data collection with the highest resemblance of words the resulting group is set as the base for the next loop round. This continues until the repetitions within the hierarchy reach the lowest entity level or the similarity score gets lower. At that point the final entity with the highest resemblance is returned and a new target word is selected.

Chapter 6

Evaluation

To test the implemented analysis tool a collection of testing material is created. Schema.org's website includes next to descriptions, properties and a hierarchical overview of all entities also examples of how to use the various semantic types. A sister web-page, which represents a small wiki on semantic schema.org information, called getschema.org contains also some usage examples of the different entities. It is therefore additionally used for the collection of testing data. In the following sections the creation of the test data collection is described next to the the tests themselves, followed by an overall evaluation.

6.1 Test data collection

As the vocabulary is quite big in its size of countable entities their examples are retrieved automatically from the entity sub pages of schema.org and getschema.org. The continuous structure within the two websites made the retrieval quick and easy. All sub pages are requested automatically to filter, extract and analyse their content. On the detailed sub pages of schema.org the examples are divided into parts of four:

- without markup,
- Microdata,
- RDFa and
- JSON-LD.

"Without Markup" represents a plain text example in HTML syntax, while "Microdata, RDFa" and "JSON-LD" use the same content with added semantics in their specific implementation. Of course JSON-LD differentiates itself from the other two implementation methods by its terminology. It is included via a JSON object within a JavaScript tag, usually at the top of a HTML document (Section 3.1.1). An example of a schema.org sub page for a trilogy of books is presented in the following code snipped.

```
<div>
  <p itemscope itemtype="http://schema.org/Book" itemid="#trilogy">
  <link itemprop="about" href="http://id.worldcat.org/fast/1020337">
    The <strong itemprop="name">Lord of the Rings</strong> is an
    <span itemprop="inLanguage" content="en">English-language</span>
    ...
  </p>
</div>
```

To retrieve the most value out of this data the information from "without Markup" and "Microdata" is extracted. At this point "RDFa" would have been a possible alternative for information extraction. The decision is mainly based on the factor that Microdata is up to now schema.org's first choice of implementation. In L2 the "itemscope" and "itemtype" attributes are included into the HTML syntax to indicate the insertion found to indicate the presence of an entity type.

The terminology of schema.org's vocabulary enables a fast and easy search through all entities, as well as the extraction of the testing data. schema.org's list of entity examples for inclusion is still incomplete. Therefore, the wiki page getschema.org, which is also partially equipped with examples, is used additionally to extract more data for the test collection. In total about a third of the vocabulary got covered through this automatic Information Retrieval. As the test data collection has a high value and importance to the evaluation of the developed system the automatically retrieved information needs to be checked manually. Some HTML examples do not follow the overall structure within the various semantic examples on the pages of schema.org and getschema.org. Span-tags are put in div-tags for example without any reasonable explanation. This causes the automatic extraction to fetch more data than necessary or the "itemprop name" to retrieve wrong information. The error rate within the collected test data is in total very low and just a few search queries were incorrectly extracted. This manual process does however not interfere with the goal of a fully automatic system as it is just used for the supervision of testing purposes.

6.1.1 Test Example Model

Schema.org's retrieved test data is saved in a Mongoose collection model (Section 5.2.1) with the following components:

- type and typeId,
- query,
- example,
- htmlexample,
- parentType and parentId.

Next to the identifying components (type, typeId) and the hierarchical information (parentType, parentId) the valuable data containing the test exam-

ple is stored. Each entity type is included with the “itemscope” and “itemtype” attributes to indicate the inclusion of semantic information. “Query” represents the target word which is in most cases the name attribute of the entity type. Within the “example” the complete test data with all by the web page provided semantic information is saved. The syntax version of it is stored as plain text in “htmlexample” to achieve a fast and easy context comparison afterwards. All data from getschema.org is saved separately in a collection for reference.

Similar to the retrieval of all descriptions and definitions for terms and entities the test data collection provides the option for an update. The constant changes and ongoing development on and behind schema.org require a certain flexibility on behalf of the system. Therefore, the possibility to update and refresh the testing data is given.

6.1.2 Expected Test Result

After a single test run the result entity types are compared to their expected single and group entity. The result with the highest confidence score is saved in a result database collection to then review, analyse and evaluate all results. Given the example input of:

```
{
    "_id" : ObjectId("55ef101b7ac188e4279458c3"),
    "type" : "MovieTheater",
    "typeId" : "558484694fd8c4181cc748db",
    "query" : "Rave Cinema",
    "example" : "HUBER HEIGHTS. The parent company of the Rave Cinema
                 in Huber Heights will renovate the 16\-screen cinema
                 and will add beer, wine and frozen cocktails to the
                 concession stand menu, Cinemark officials announced
                 Thursday. Cinemark Holdings, owner of the Rave
                 Cinemas Huber Heights 16 at 7737 Waynetowne Blvd.,
                 said the multiplex will offer Luxury Lounger recliners
                 in all auditoriums, an updated lobby and new
                 concession stand and bar area ...",
    "parentType" : "Place",
    "__v" : 0
}
```

the system will analyse the query (with the expected type found in attribute “type”) and it’s surrounding content (given in the “example” attribute). Their most valuable identifiers are calculated, existing inherited hypernyms are tried to be detected and the results are compared to all entity identifiers and their hierarchical structure to identify the entity type with the highest similarity value.

A successful and therefore passed test is expected to have matching values between its single type entity ID and result type ID as well as in its group type and result group type.

```
{
  "typeId" : "558484694fd8c4181cc748db",
  "resultTypeId" : "558484694fd8c4181cc74968",
  "singleType" : "MovieTheater",
  "resultType" : "MovieTheater",
  "groupType" : "Place",
  "confidence" : 89,
  "query" : "Rave Cinema",
  "_id" : ObjectId("55f0987d6f38ae08107aee2f")
}
```

In the given example the identical ids are displayed with a calculated confidence value of 89, which indicates a high similarity between the calculated identifiers for the term “Rave Cinema” and the entity type “MovieTheater”.

6.2 Testing environment

Most implementations of testing environments are focused on clean code and an evaluation if a test returns the expected results. The tests for this particular system are used to check how many of all entities are identified correctly by their parent group and as a single type. As this is more specific and focused on the terms of usage a small testing environment is created. After the request of all collected information for identifiers and tests the system runs through each test query before saving the results.

6.2.1 AYLIEN’s Text Analysis API

As mentioned before, AYLIEN’s Text Analysis API (Section 3.4.2) offers already a detailed and well elaborated tool to identify a low but steady number of entities within a plain piece of content. Their API is therefore chosen as a base for the developed system and testing tool so that the unsupervised system does not have to focus on these already worked on entity types.

As already described in Chapter 3, AYLIEN includes the analysis and detection of 42 entity types with a supervised approach. These do not match up one hundred percent with the mostly used schema.org types (Figure 3.4). Due to the high precision in their results for the detection of these distinct entities the API is used to identify them. For all others the developed system jumps in and classifies them according to the preprocessed entity data sets.

The API is easily included via the provided NodeJS SDK and requires only an application ID and key for initialisation which are received after registering the application. Various methods for calling different API endpoints are available and *Concept Extraction* is in use for this particular system. After calling the API on a target word a list of concept types is sent back. These might contain semantic types of DBpedia schema.org and/or Wikidata. Subsequently, the developed system starts its own analysis and testing,

but only if the extraction tool does not detect any schema.org entity type that would fit the target word.

6.2.2 Test Query Representation

The actual testing starts as soon as the Concept Extraction tool does not find any entity match-up for the target word. A test begins with the retrieval of more specific data concerning the test query. The system already knows some information through the context around the target word but that environment does not always say a lot about it. Supervised AI systems use their created data-collection on word- and term-usage within different types of context to identify the type and sense of a query. Unsupervised techniques require external data-sets to come to conclusions. Therefore more data needs to be collected and run through comparisons during run-time. The inclusion of dictionaries such as Wolfram Alpha and WordNet are used in the first approach of testing, as well as Bing Search and Wikipedia for target query Information Retrieval.

At this point the Information Retrieval via Bing Search is done carefully as the first results are mostly Wikipedia and possibly the schema.org or getschema.org websites with the examples that were extracted. The API for Bing Search offers the possibility to skip the first couple of results so that redundant information is avoided. This is done to have a perfect match for the result if the example sub page is found in the search request. In the same procedure as for the entity identifier selection the collected data is analysed to extract their most important terms and words. These are used additionally to the already known content around the search query and the target word itself and create the test query representation for the upcoming comparison. To follow the same procedure as for the entity identifiers, the collected data needs to be stemmed before the comparison in the next step.

6.3 Testing Categories

During development multiple tests are used to identify the best techniques to collect and determine the best suitable identifiers for all single and group entities. Subsequently to various changes in methods and heuristics a first evaluation is taken. To identify the strengths and weaknesses of the developed system three testing categories are identified to compare the respective results.

6.3.1 Stemmed Group Test

To achieve the best results not only one but multiple vector space models are used to calculate the highest confidence level possible within the stemmed

group test (SG). In this particular round of testing the following data sources are chosen to determine the best suited group identifiers:

- query (term and context),
- data-sets (group and single entities),
- WordNet (definitions and synonyms),
- WolframAlpha (definitions and synonyms) and
- Bing (rich snippet in search result).

A heuristic on the top terms from the listed dictionaries and search engines is set via TF-IDF. Another one is used to increase the confidence level on higher similarities between the test query and data-set of the entity group in focus as well as between the search engine results of both components.

The main comparisons and calculations are divided into four main parts: Group data-set, Bing, WordNet and WolframAlpha. Also a changed weighing scheme of nouns in a closer window to the query in focus is used to increase their importance for sense-disambiguation. This is essential for the detection of specific single entities in the lowest hierarchy branch. These entities differentiate themselves in small details (e.g. childcare vs. kindergarten), which are difficult to detect from a wide range of identifiers. That's why the context is important in various cases to detect the correct entity type.

6.3.2 Lexical Knowledge Base Test

Lexical knowledge-bases are used as a base for this approach. The heuristics are set via the Lesk Algorithm which states that terms are defined as disambiguated if their description in dictionaries have the greatest word overlap count. Therefore, the inherited hypernym structure between the query and each single entity is compared via the use of vector space models. Furthermore, similarities or even equalities between the dictionary and vocabulary hierarchies is used as a weighting factor to determine word senses. More complex and deeply structured dictionaries, such as WolframAlpha and WordNet, offer connections and inheritance information for each entered term. This feature is used to identify the overall entity group of search queries to then determine more precisely the ideal single entity type. The data collected for each entity type is widened by the addition to all entity names on top of the list of identifiers.

Furthermore, the entity names are compared to each other so that the children entities won't include the identical identifiers as their parents, in the sense of having BodyOfWater as a parent and *LakeBodyOfWater* as a child removes *BodyOfWater* from the child to make the specification in lower branches more unique. All entity names use specific word combinations to differentiate them from each other (e.g. SportsOrganization). These simple words have a high probability to collect useful results in dictionary search.

Table 6.1: Test Results

	SE	LKB	SG
Consumed time per entity (in sec)	2,14	7,28	5,14
Passed Single	29,62%	30,07%	35,77%
WSD	29,62%	75,12%	67,31%

In addition, the inheritance information between all results is compared to identify the best suitable definition. This avoids the wrong disambiguation of words through the multiple options of definitions in the queried result.

6.3.3 Single Entity Test

While the tests from Stemmed Group and Lexical Knowledge Base are focused on pre-processed data per grouped parent entity, the single entity test (SE) is based on the collected data per entity type. This approach differentiates a term's word-senses by dividing the queries per word-type. To achieve the best results possible all queries are divided into four categories which are to be identified by the intelligence of the system:

- proper nouns (CKBs),
- nouns (LKBs),
- verbs (LKBs) and
- alternative values (SER).

Most queries and semantically described terms have a high probability to represent proper nouns. Further information and precise data for this category is found in collaborative knowledge bases (CKBs) and via search engine results (SER). All action entities, specifically for the schema.org vocabulary, are grouped within the verb category. Nouns represent a rather small group in comparison to proper nouns but are still used regularly for semantic representations of websites. Valuable data for regular nouns is extracted from lexical knowledge bases (LKBs).

The most challenging group to identify as single entities but the easiest to detect as group entities are the remaining alternative values. These include numbers, media objects, code snippets or website components, such as menus and site classifications. They differentiate themselves from all other entities by not retrieving any valuable information from all described sources.

6.4 Results

The described testing approaches (single entity test, lexical knowledge-base test, stemmed group test) are evaluated in three categories: performance, passed single tests and word-sense disambiguation, displayed in Table 6.1.

Table 6.2: Failed Detection Distribution

Groups	**Failed tests in % per individual group test**
Action	–
BroadcastService	0,00 %
CreativeWork	36,36 %
Event	9,09 %
Intangible	84,09 %
MedicalEntity	16,00 %
Organization	12,00 %
Person	0,00 %
Place	6,98 %
Product	0,00 %

The performance category shows the biggest differences in its result values. SE testing is by far the fastest approach with a time of 2,14 seconds to calculate the highest confidence level. LKB and SG achieve with over one hour of testing time for around 300 queries and 700 poor performance values.

6.4.1 Lexical Knowledge Base Test Result

The inclusion of lexical knowledge bases leads overall to impressive identification results. With a precision value of 75,12% in word-sense disambiguation the introduced comparison of inheritance and hierarchical structures lead to better results than the best unsupervised system in SemEval2007 (69%). A closer look into the failed test results, and the missing 25% to reach one hundred indicates that the result values are possibly still lower than they could be. Certain entities within the different hierarchical structures in the used LKBs do not always reflect the same inheritance as the vocabulary of schema.org.

In Table 6.3 the inherited hypernyms of the terms “sea” and “church” in the vocabularies of schema.org and WordNet are presented. The example for term “sea” shows a small difference in the parental types as schema.org offers a higher complexity in its differentiation between children-types. But the main focus is set on the matching parent type for “BodyOfWater”. This inheritance similarity is used for the grouping process of this approach. On the right side of the table the example for the term “church” is displayed. Here, the differences between the two vocabularies are critical. The entities’ parent types belong to two completely different branches and cause therefore a failed test for the “church” entity.

Table 6.2 displays a more precise and detailed listing of the tests that failed the grouping on the first level. All tests from the categories “Broad-

Table 6.3: Inheritance Type Comparison (from single to parent entity)

Sea		**Church**	
schema.org	*WordNet*	*schema.org*	*WordNet*
SeaBodyOfWater	Sea	Church	Church
BodyOfWater	BodyOfWater	PlaceOfWarship	Religion
Landform	Thing	CivicStructure	Institution
Place		Place	Organisation

castService", "Person" and "Product" are classified correctly in their word-senses. The group of "Person" passed without any surprise, as AYLIEN claims to have perfected their detection for this specific entity type. "BroadcastService" is a very small group as it consists of only one entity. Therefore the most success is read out of all passed tests for "Product". With only around 7% to 16% failed tests the results for "Event", "MedicalEntity", "Organization" and "Place" are seen as a great accomplishment as well. The failure of 36,36% on behalf of "CreativeWork" is on the other hand a big disenchantment. Thinking of creatively created documents they should be easily identified as they differentiate themselves a lot from all other entities. "Intangible" represents everything that does not fit in any other category which makes the high percentage of failed tests not as disappointing.

Performance

The loop to identify the best suitable parent group and single entity takes on a lot of time through the API calls of the dictionaries WordNet and WolframAlpha. As a result each test loop is done within a couple of seconds which adds up to a total testing time of over thirty minutes per dictionary. Although the test results are satisfactory the entity determination should not take as much time as it does at this point. Depending on the user and project of the system the identification of all entities could take up a couple of hours.

Because of the fact that this part of the system is not user friendly, the calls to WolframAlpha and WordNet are cut out of the search query Information Retrieval in SG testing and replaced by the challenge of improving preprocessing. The already retrieved data for single and group entities needs to be more precise and include better identifiers and heuristics to enhance the now dropped results for passed single tests of 23,79% and 45,35% correct identifications in WSD.

Table 6.4: Failed Detection Distribution

Groups	Failed tests in % per individual group test	LKB results
Action	–	–
BroadcastService	0,00 %	0,00 %
CreativeWork	49,06 %	36,36 %
Event	75,00 %	9,09 %
Intangible	51,35 %	84,09 %
MedicalEntity	28,00 %	16,00 %
Organization	53,57 %	12,00 %
Person	0,00 %	0,00 %
Place	3,16 %	6,98 %
Product	83,33 %	0,00 %

6.4.2 Stemmed Group Test Result

After the removal of WordNet and WolframAlpha from the run-time analysis the result values for passed tests decreased reasonably. Identical or related words and terms deliver usually similar results in their knowledge base requests. These are now missing with the consequence of an increase in failed tests. This calls out for improvement in pre-processing and the inclusion of alternative techniques to determine identifiers for group and single entities. To improve the data created in preprocessing a closer look at all test results is taken. Some of these examples actually achieved high similarity values but did not score as well as they could. One of the reasons being that the terms did not result in a high affinity score with the selected identifiers. Words in plural are already reduced to their singular form to achieve a higher similarity rate but other grammatical changes are not prioritised. This problem is taken care of with a more precise inclusion of stemming. Another observation is made on the queries themselves. They are mostly proper nouns describing the art work in focus. This makes the use of search engine results, CKBs and the analysis of the surrounding context more attractive than dictionaries. Therefore, a shift in heuristics through the inclusion of higher weighting schemes is included.

In Table 6.4 the new results per grouped entity are displayed as well as the values from the LKB testing for comparison. The biggest difference for all main types is the change in passed semantic identification for “Product”. Almost all queries in this group were categorized as “Place” entities. The highest weight which lead to a wrong categorization is the location information within the context and window-nouns around the term in focus. A change in heuristics to weigh the context for proper nouns differently resulted in worse results for all other groups. Therefore, the used heuristics

were not changed. A closer look into the failed group tests of "CreativeWork" in SG testing makes more challenges concerning the hierarchical structure of all entities visible that still cause problems in determining the correct entities. The type "MedicalScholarlyArticle" failed its test as it appears as a child-entity in the group "MedicalEntity" as well as in "CreativeWork". Due to a higher similarity score with the medical entity group the query is categorized in the wrong but also semantically correct group. These failed tests lead back to the missing LKB calls and their inheritance information.

6.4.3 Single Entity Test Result

Single entities present themselves in various ways, meaning to have proper nouns as well as verbs, regular nouns and alternative values. Especially these alternative values are difficult to detect because the collection of identifying terms in an unsupervised manner is made challenging. One of them being a train reservation number, such as *#AB3XY2*. These types of queries are too specific in their value that neither dictionaries nor collaborative knowledge bases are able to collect identifying data for them. The only possibility to gather additional information for these types of values are search engine requests. A good amount of the engine results for the set query consist of ticket information for either flights, events, trains or other types of tickets. Through this information and the analysis of the surrounding content of the query in focus helps to identify the type of ticket but does not result in all cases with the correct entity type.

In Table 6.5 passed tests for various types of tickets are displayed. Comparing the queries of these tests makes their uniqueness and the similarity between them obvious. The content around the example of the FlightReservation type is as follows: "Reservation #RXJ34P Passenger: Eva Green Flight: United Airlines Flight 110 Operated By: Continental Airlines Departing: San Francisco Airport (SFO) Arriving: John F. Kennedy International Airport (JFK)."

Hereby the context does say a lot about the query. Therefore it requires more weight for the calculation of the confidence level to determine the correct entity type. Table 6.6 displays failed single tests for alternative values. These have most likely not been identified with the correct entity type because of their highly weighted surrounding content. In the example "Original: Reservation #E123456789 under name: John Smith Foo Fighters Concert AT&T Park 24 Willie Mays Plaza San Francisco, CA 94107 Ticket #849646815 Section: 101 Row: A Seat: 12" it is visible that the context is mostly talking about an event, while the query in focus is concerned about the ticket which is taking up only a small part of the whole text.

Table 6.5: Alternative Value Evaluation - Passed Single Tests

Query	Type	Confidence
#OT12345	FoodEstablishmentReservation	18
#E123456789	EventReservation	6
#RXJ34P	FlightReservation	8
#546323	TaxiReservation	8
#AB3XY2	TrainReservation	11

Table 6.6: Alternative Value Evaluation - Failed Single Tests

Query	Type	Result Type	Confidence
#849646815	Ticket	EventReservation	6
#184D93KL	LodgingReservation	Car	5
#36279808A	RentalCarReservation	Car	5

6.5 Overall Evaluation

The overall results showed improvement in their precision values using the following adjustments:

- Including inherited hypernyms out of WordNet and comparing it to the hierarchical inheritance structure of schema.org.
- Increasing the weighting on higher similarity scores between the vector space models of window nouns and entity identifiers.
- Excluding news articles as well Wikipedia pages in search results, while only collecting data from the top three rich snippets.
- Determining the most valuable identifiers via the calculation of TF-IDF and subsequently using a stemmer on these terms.
- Using only the first paragraph of a Wikipedia article as opposed to the whole article leads to better precision but decreases recall.

6.5.1 Specific Vocabulary and Entity Observations

When analysing the vocabulary of schema.org in more detail to study the structure of a semantic hierarchy, some characteristics are standing out to be the main focus for the first part of testing. “Action” represents a very specific group type of entities. These entities describe movement and physical actions of humans, such as cook, win, agree or subscribe.“John communicated with Steve” represents a usage example for one of these actions called “communicate” shown on schema.org’s website. The detection of these types of entities differentiates itself in multiple aspects. One of them being that the target word or query refers always to a verb, while all other entities refer to

nouns and proper names. The various actions are conclusively saved in their most basic stemmed form within the database to then be easily detected by the system. Another discrepancy to all other entities lies in their form of use. Schema.org does not deliver a way of including these types of entities other than with JSON. This leads to the conclusion that "Action" entities are still to be worked on and might be included differently to all other entity siblings.

From an Artificial Intelligence point of view this explicit group does create a challenge as well. The vocabulary is meant to be variable and replaceable. To create a global system that works for all semantic vocabularies appears to be a not reachable but at the same time still obtainable goal. Depending on the objective one has set and the sacrifices that are willing to be taken. The out-comings after testing are still most likely to change in relation to the vocabulary because of the inconsistency and high number of variables that can't be all covered. The system could analyse each entity first and define their type of word before identifying other terms as their type. In the first approach of designing the system this step was taken but ending up with poor results. The single descriptive sites for every entity do not contain enough information to distinguish one type from another. The already depicted problem of having missing properties and attributes make the automatic and unsupervised collection of correct data to describe entities unattainable. After pre-analysing the vocabulary for word types and unsatisfying results the entity group "Action" is excluded in the first couple of tests.

Chapter 7

Conclusion

Via word-sense disambiguation the true meaning of terms in context is identified in a computational manner. The Semantic Web represents combined with various vocabularies a tool for machines to understand and interpret words and terms correctly. But the vast amount of unstructured data within websites does still create a barrier for systems to fully understand user-generated content and identify a word's true meaning. The areas of Artificial Intelligence and Semantic Text Analysis enable in combination with semantic vocabularies different methods and approaches to transform unstructured data into structured content. This thesis displays the possibilities and restrictions of a semi-supervised system to evaluate a context's meaning and determine entities to include as well as to enrich it semantically.

The comparison and evaluation of all testing approaches ranged from a stemmed grouping method (SG), over an all single entity detection (SE), to a lexical knowledge based and inheritance approach (LKB). LKB testing, which introduces the use of inherited hypernyms in dictionaries, achieved the best results for word-sense disambiguation with a 75,12% precision value and outperformed the best unsupervised system in SemEval2007 (69%). The created system represents a solid base for further development in the areas of Natural Language Processing, Artificial Intelligence, Semantic Text Analysis and unsupervised Word-Sense Disambiguation to identify important information within unstructured content and include semantic information for machines to interpret and understand. Through the avoidance and exclusion of huge data collections to describe each entity, external knowledge bases are used to collect valuable data and differentiate entities from each other. In combination with hierarchical information on parent and child entities the sense of words is identified in an early stage of entity detection. This reduces the chance of wrongly defining a word's meaning to correct identification in three out of four words.

7.1 Outlook

After the analysis of semantic vocabularies and the test results of the developed system it is to be concluded that there are still areas left that need improvement. The described vocabulary of schema.org is yet to be completed in its hierarchy and entity definitions. The sub-branch for medical information is already represented with close detail but other areas need more specific entities to describe proper nouns correctly. A solution to this lack of data representation is the inclusion of separately created vocabularies following the guidelines of schema.org. This enables the incorporation of a wider range of entity types that are defined through attributes and properties created by professionals in the various fields. An AI system with semi-supervision for semantic analysis is to be developed which is divided into several sub-parts for more specific fields to automatically detect detailed entities and identify them within plain and unstructured content.

Appendix A

Contents of the DVD-ROM

Format: CD-ROM, Single Layer, ISO9660-Format

A.1 PDF files

Pfad: /

_Thesis.pdf . . . . . Master thesis

Pfad: /online-literature

AaronBradley.pdf . . . . [29]
RDFModel.pdf . . . . . [30]
RDFaLite.pdf . . . . . [31]
MikrodatenWortschatz.pdf [32]
SnowballStemmer.pdf . [33]
PortalStemmer.pdf . . . [34]
PayLevelDomain.pdf . . [35]
WebDataCommonsDataSets.pdf [37]
WebDataCommonsDataSetsDataCrawl.pdf [36]
WhatIsSchema.pdf . . . [38]

A.2 Source Code

Pfad: /code

/app/run-tests.js . . . . Testing environment to connect to developed system.

A.3 Other

Pfad: /images

*.jpg, *.png . . . . . Original Images
*.ai . . . . . Original Adobe Illustrator Files

References

Literature

[1] John Allsopp. *Microformats: Empowering Your Markup for Web 2.0*. New York City, USA: Friends of ED, Mar. 2007 (cit. on p. 8).

[2] José Luis Ambite et al. *Automatically constructing Semantic Web Services from online sources*. Tech. rep. Marina del Rey, CA: University of Southern California Information Sciences Institute, 2009 (cit. on p. 18).

[3] Ping Chen et al. *A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge*. Tech. rep. Boulder, Colorado: Association for Computational Linguistics, 2009 (cit. on p. 13).

[4] Timothy Chklovski and Rada Mihalcea. *Building a Sense Tagged Corpus with Open Mind Word Expert*. Tech. rep. Cambridge, MA and Richardson, TX: MIT Computer Science Artificial Intelligence Laboratory, and UT Dallas Department of Computer Science, 2002 (cit. on p. 13).

[5] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Language, speech, and communication. MIT Press, 1998 (cit. on p. 26).

[6] Michael Lesk. *Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone*. Tech. rep. Morristown, NJ: Bell Communications Research, 1986 (cit. on pp. 12, 13).

[7] Dekang Lin. *Sense Ambiguity*. Tech. rep. Manitoba, Canada: Department of Computer Science University of Manitoba, 1990 (cit. on p. 13).

[8] Diana McCarthy et al. *Finding predominant word senses in untagged text*. Tech. rep. Brighton, UK: Department of Informatics, University of Sussex, 2004 (cit. on p. 14).

[9] Rada Mihalcea. *Using Wikipedia for Automatic Word Sense Disambiguation*. Tech. rep. Rochester, NY: North American Chapter of the Association for Computational Linguistics – Human Language Technologies, 2007 (cit. on p. 2).

[10] Rada Mihalcea. *Using Wikipedia for Automatic Word Sense Disambiguation*. Tech. rep. Rochester, NY: Association for Computational Linguistics, 2007 (cit. on p. 13).

[11] Peter Mika and Tim Potter. "Metadata Statistics for a Large Web Corpus". In: *Linked Data on the Web*. Ed. by Christian Bizer et al. Lyon, Apr. 2012 (cit. on p. 18).

[12] Dan Moldovan and Vasile Rus. *Explaining Answers with Extended WordNet*. Tech. rep. Dallas, TX: Department of Computer Science and Engineering Southern Methodist University, 2001 (cit. on p. 13).

[13] Raymond J. Mooney. *Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning*. Tech. rep. Austin, TX: Department of Computer Sciences University of Texas, 1996 (cit. on p. 11).

[14] Roberto Navigli and Mirella Lapata. *Graph connectivity measures for unsupervised word sense disambiguation*. Tech. rep. Hyderabad, India: IJCAI International Joint Conference on Artificial Intelligence, 2007 (cit. on p. 12).

[15] Roberto Navigli and Simone Paolo Ponzetto. *Joining Forces Pays Off: Multilingual Joint Word Sense Disambiguation*. Tech. rep. Jeju, Korea: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012 (cit. on p. 15).

[16] H.T. Ng, B. Wang, and Y.S. Chan. *Exploiting parallel texts for word sense disambiguation: An empirical study*. Tech. rep. Sapporo, Japan: Association for Computational Linguistics, 2003 (cit. on p. 13).

[17] Adrian Novischi, Muirathnam Srikanth, and Andrew Bennett. "LCC-WSD: System Description for English Coarse Grained All Words Task". In: *Proceedings of the 4th International Workshop on Semantic Evaluations*. Ed. by Eneko Agirre, Lluis Marquez, and Richard Wicentowski. Prague: Association for Computational Linguistics, June 2007 (cit. on pp. 1, 2, 12, 13).

[18] Stephen Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF". In: *Journal of Documentation* 60.5 (2004) (cit. on p. 31).

[19] Karen Spaerck Jones. "IDF term weighting and IR research lessons". In: *Journal of Documentation* 60 (2004) (cit. on p. 5).

[20] Francisco Viveros-Jiménez, Alexander Gelbukh, and Grigori Sidorov. *Simple Window Selection Strategies for the Simplified Lesk Algorithm for Word Sense Disambiguation*. Heidelberg: Springer-Verlag, 2013 (cit. on p. 14).

[21] Ellen M. Voorhees. *Using WordNet to Disambiguate Word Senses for Text Classification*. Tech. rep. Rochester, NY: Special Interest Group on Information Retrieval, 2007 (cit. on p. 2).

[22] David Yarowsky. "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods". In: *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*. Ed. by Hans Uszkoreit. Saarbrücken, Germany: Association for Computational Linguistics, June 1995 (cit. on p. 13).

Online sources

[23] URL: https://schema.org/ (visited on 04/02/2015) (cit. on p. 8).

[24] URL: http://www.thefreedictionary.com/synset (visited on 04/02/2015) (cit. on p. 14).

[25] URL: http://www.searchengineshowdown.com/defs/stop.html (visited on 04/02/2015) (cit. on p. 14).

[26] URL: https://www.similartech.com/categories/schema (visited on 08/22/2015) (cit. on p. 21).

[27] URL: http://docs.aylien.com/docs/concepts (visited on 04/04/2015) (cit. on p. 22).

[28] URL: http://www.collinsdictionary.com/dictionary/english/stemming (visited on 05/04/2015) (cit. on p. 32).

[29] *Basic definitions related to schema.org by Aaron Bradley*. URL: https://plus.google.com/106943062990152739506/posts/8X4CfZGiUhB (visited on 07/20/2015) (cit. on pp. 9–11, 65).

[30] *Making Statements About Resources - The RDF Model*. URL: http://www.w3.org/TR/WD-rdf-syntax-971002/ (visited on 07/14/2015) (cit. on pp. 10, 65).

[31] *RDFa Lite 1.1 - Second Edition*. URL: http://www.w3.org/2010/02/rdfa/sources/rdfa-lite/Overview-src.html (visited on 03/29/2015) (cit. on pp. 10, 65).

[32] *schema.org in den Suchergebnissen: ein wertvoller Mikrodaten-Wortschatz*. URL: http://www.searchmetrics.com/de/news-und-events/schema-org-in-den-suchergebnissen/ (visited on 04/14/2015) (cit. on pp. 18, 65).

[33] *Snowball: A language for stemming algorithms*. URL: http://snowball.tartarus.org/texts/introduction.html (visited on 05/04/2015) (cit. on pp. 32, 65).

[34] *The Porter Stemming Algorithm*. URL: http://tartarus.org/~martin/PorterStemmer/index.html (visited on 05/04/2015) (cit. on pp. 32, 65).

[35] *Vocabulary Usage by Pay-Level Domain.* URL: http://webdatacommons.org/structureddata/vocabulary-usage-analysis/ (visited on 04/16/2015) (cit. on pp. 18, 65).

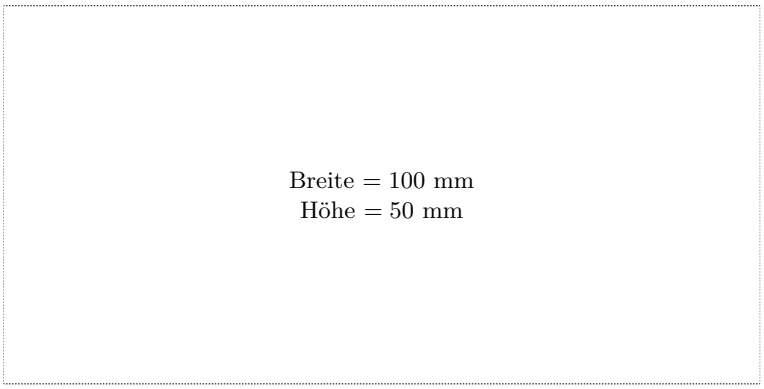
[36] *Web Data Commons. Extraction Results. Common Crawl Corpus - Trends 2012 to 2014.* URL: http://webdatacommons.org/structureddata/index.html#trend-2012-2014 (visited on 07/18/2015) (cit. on pp. 8, 65).

[37] *Web Data Commons - RDFa, Microdata, and Microformat Data Sets.* URL: http://webdatacommons.org/structureddata/ (visited on 04/16/2015) (cit. on pp. 19, 65).

[38] *What is Schema.org? By Phil Barker and Lorna M. Campbell.* URL: http://publications.cetis.org.uk/2014/960 (visited on 04/14/2015) (cit. on pp. 16, 17, 65).

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —