Cologne University of Applied Sciences

07 Faculty of Information, Media and Electrical Engineering
Institute of Communications Engineering

# Design and Implementation of a Knowledge Acquisition System for Automated Ontology Synthesis; based on Natural Language Processing including Methods from Computational Intelligence

B. Sc. Andreas Schwenk

Submitted in partial fulfillment of the requirements for the Master module
"Research Project"

| | |
|---|---|
| Student: | **Andreas Schwenk**<br>Bachelor of Science in Information Engineering |
| Matriculation number: | 11069755 |
| Program: | Information Engineering (Master's Program) |
| Supervisor: | Prof. Dr. phil. Gregor Büchel |
| Handover date: | 2014-05-06 |

The presented report is submitted for the partial fulfillment of the requirements for the master module “Research Project” [1].

I hereby declare that the paper submitted was in all parts exclusively prepared on my own. All other resources or other means, than those explicitly refereed to, have not been utilized. All implemented fragments of text, employed in a literal and / or analogous manner have been marked as such.

Windeck and Cologne in May 2014, Andreas Schwenk

[1] “Research Project ” is a 10 ECTS Master Course (Program: Information Engineering) at Cologne University of Applied Sciences (=: CUAS). The course furthermore consists of a practical part and a presentation of the final results.

# Contents

**Abstract**

A knowledge acquisition system has been designed and implemented. The system is capable of extracting information from a corpus, consisting of natural language based – and thus unstructured – texts about a given knowledge domain $D$. This is achieved by a collaboration of algorithms from Computational Intelligence and Computational Linguistics (both subsets of the Artificial Intelligence). The results may be used later as a basis for a Knowledge-Based System (=: KBS).

The presented pipeline is subdivided into “Preprocessing” and “Information Extraction”. The former consists of a syntactic and a semantic analysis, i.e. it disassembles the text into (meta-) tokens, enriches it with semantic relations, determines the word classes, the word frequencies and predicts the tense per phrase. A new Neuro-Fuzzy based approach for hierarchical parsing of a dependency grammar was developed. The preprocessing part is supported by a set of web resources. The second subsystem consolidates the raw data to finally generate a structured knowledge representation in form of an ontology $O' \subseteq O = (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \mathcal{A}0)$. Instruments are word frequency analysis, lexical databases and dependency grammar analysis.

The chosen testing domain $D :=$ “The Structure of The Universe” evaluates the applied methods.

KEYWORDS: Artificial Intelligence, Artificial Neural Networks (Supervised Learning), Computational Intelligence, Crisp Logic, Dependency Grammar Parsing, Fuzzy Logic, Information Extraction, Information Retrieval, Knowledge Based Systems, Lexical Databases, Natural Language Processing, Ontologies, Processing of Linguistic Signs, Semantic and Syntactic Analysis, Unstructured Text Processing, Word Frequency Analysis.

## List of Abbreviations / Acronyms

| | |
|---|---|
| $\mathcal{A}0$ | Axioms |
| ANN | Artificial Neural Network |
| ASCII | American Standard Code for Information Interchange |
| BNC | British National Corpus |
| $\mathcal{C}$ | Concept |
| CI | Computational Intelligence |
| CSV | Comma-Seperated Values |
| D | Domain |
| DG | Dependency { Grammar \| Graph } |
| DOM | Document Object Model |
| DoT | Degree of Truth |
| EBNF | Extended Backus–Naur Form |
| ERD | Entity Relationship Diagram |
| FL | Fuzzy Logic |
| FLE | Fuzzy Logic Editor |
| FLS | Fuzzy Logic System |
| FRS | Fuzzy Rule Synthesis |
| GUI | Graphical User Interface |
| $\mathcal{HC}$ | Hierarchy of Concepts |
| HTML | Hypertext Markup Language |
| HYP | Hyperonym (=: Hyernym)[2] |
| KBS | Knowledge-Based System |
| KDB | Knowledge Data Base |
| LEX | lexeme |
| LT | Linguistic Term |
| LV | Linguistic Variable |
| MLP | Multilayer Perceptron |
| NLP | Natural Language Processing |
| NP | Nominal Phrase |
| O | Ontology |
| OE | Ontology Extraction |
| OWL | Web Ontology Language |
| PMF | Probability Mass Function |
| $\mathcal{R}$ | Relation |
| R | Rule |
| RDF | Resource Description Framework |
| RDFS | RDF-Schema |
| STK | Semanticized Token |
| SYN | Synonym |
| TS | Training Set |
| UML | Unified Modeling Language |
| VP | Verbal Phrase |
| XML | Extensible Markup Language |

[2] Often abbreviated to "hr". In some cases "hyp" means "hyponym". We always use the definition: "hyp := hyper(o)nym" in this document.

# Chapter 1

# Introduction

## 1.1 Motivation

The amount of information increased exponentially since the invention of writing systems[1]. To overcome this (metaphorical derived term:) *Information Explosion*, “techniques to gather knowledge from an overabundance of electronic information” [Wik14a] have been developed.

An example application field for automated information retrieval (We call it here: Knowledge Acquisition; refer also to the relation of “Data Mining”) is e.g. an autonomous system, i.e. a robot, that communicates with human beings. Whereas it is not clear if artificial intelligence will ever reach the power of the human brain (either in terms of logic or creativity), methods of information gathering to build up a knowledge base, is substantial. Beneath this – more or less science fiction scenario – we need to supply existing and intensely used systems like software agents with information, i.e. a formal structure of knowledge representation for reasoning.

Information can be either stored as *structured*[2] or *unstructured* data, or in a hybrid form – so called *semi structured* data – e.g. XML[3]. The former is homogeneously formed and therefore rather simple to parse. In this project, we will focus on unstructured data, i.e. natural language texts. Reasons for this decision is the respect to human evolution and thus, the high-quantitative availability of natural language text sources. The main challenge in *Natural Language Processing* (=: NLP), that is in this case parsing and machine-learning natural language texts, is *uncertainty*. We can only extract features in form of facts up to a certain *probability*.

To give a perspective to parts of this work, we list the key topics from the science of Natural Language Processing[4]. The terminology is taken from [CFL12] an can be interpreted as a partial quantity of the entirety of this knowledge domain. The list has no certain structure and topics that correlate with this project are tagged with “(∗)”:

- Automatic summarization (∗)
- Machine Translation
- Natural language generation
- Natural language understanding (∗)
- Parsing (∗)
- Question Answering
- Relationship extraction (∗)
- Sentence breaking (∗)
- Speech recognition and segmentation
- Topic segmentation (∗)
- Word segmentation (∗)

[1] Books losses in the late antiquity left out. Compare e.g. to [Joh65].

[2] Relational databases; tables.

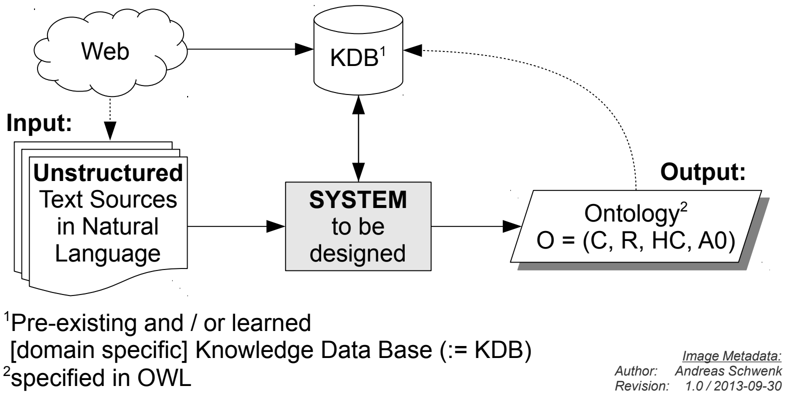
[3] XML := Extensible Markup Language.

[4] NLP := { Computer Science ∪ Artificial Intelligence (=: AI) ∪ Linguistics. }; deals with Human-computer interaction.

## 1.2 Objective

The main objective of this work is concretized in figure 1.1.

Figure 1.1: Objective – Simple Overview


Web
KDB[1]
Input:
Unstructured Text Sources in Natural Language
SYSTEM to be designed
Output:
Ontology[2] O = (C, R, HC, A0)
[1]Pre-existing and / or learned [domain specific] Knowledge Data Base (:= KDB)
[2]specified in OWL
Image Metadata:
Author: Andreas Schwenk
Revision: 1.0 / 2013-09-30

The system boundary is defined as follows: First, a set of unstructured text sources $t$ in natural language (=: corpus) is read. The system extracts data[5] and stores it in a *Knowledge Data Base* (=: KDB) for intermediate persistence. The latter may be partial exist *a priori* – e.g. from previous system runs – and thus it can be reused in subsequent steps to get better convergence, i.e. to increase knowledge and to reduce uncertainty. E.g. we will later see that a high quantity of text sources improves the detection of keywords, i.e. concepts. The output is structured as an (formal) ontology $O$ and is calculated via a hypothetical function $f$ for the corpus $C = \{t_1, \ldots, t_N\}$ as follows:

$$\begin{array}{l} (1.)\ O_0 := \emptyset \\ (2.)\ \forall_{i \in [1,N] \in \mathbb{N}}\ :\ O_i\ :=\ f(t_i,\ \text{KDB},\ O_{i-1}) \end{array} \tag{1.1}$$

with $O := (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \mathcal{A}0) := (\text{Concepts}, \text{Relations}, \text{Hierarchy of Concepts}, \text{Axioms})$, $t_i$ the currently processed text, $O_i$ the current Ontology estimation, and $N$ the number of system runs. Furthermore we only consider $O' := (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \emptyset) \subset O$.

## 1.3 Key Aspects to be Considered

The following guiding questions must be considered over the entire design process. They implicitly define the multifacetedness of natural language processing:

- How can a text written in a *natural* language be parsed? What is the difference to process *formal* languages?
- How can semantics and semantic fields – e.g. polysemy – be extracted?
- Which models and methods exist to represent *uncertainty*? Which parameters, i.e. concrete implementations, can be applied in the field of NLP?
- How can conclusions be drawn?
- Is it possible to effectively learn complex relationships (including semantic relations) from natural language without having a priori knowledge in some kind of databases (=: DBs)? If not; which kind of databases will be needed, and do these DBs have to be domain-specific?
- How can concepts be determined from a set of substantives?
- How can the extracted knowledge be transformed into a formal ontology $O = (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \mathcal{A}0)$?

[5] The generic term “data” will be explicitly specified below.

## 1.4 Determination of a Knowledge Domain for Evaluation

To verify the results[6], the developed system must be tested with a knowledge domain $D$. As stated above, the system input must be a corpus of natural language based and unstructured texts in English. We have chosen $D :=$ "The Universe and its structures" as a concrete evaluation domain. Main topics may be { orbits: planets, stars, . . .; galaxies, galaxy groups, galaxy clusters, filaments, . . .; discoverers; history; . . . }. Advantageous terms for the chosen domain can be summarized as first, that a very large base of online text sources is freely available. Since the domain belongs to the natural sciences, the text sources have (presumably) a very factual and objective character. Semantic fields may be more easy resolved, due to their minor frequency in occurrence; e.g. homonyms are less frequent in contrast to text about human sciences. Despite, the usual characteristics of natural language is retained, as information may be expressed in a very heterogeneous way. Thus the general demand for a system that is resistant to uncertainty remains. Due to an online-behavior of the system, the domain could be replaced by another domain or expanded at any time, for further fine-tuning aspects.

## 1.5 Contents

The text at hand is divided into two parts: While part one describes the theory, concepts and evaluation, part two consists of the listing of (nearly) all source codes[7].

The first chapter gives an overview about the major project goals and briefly enumerates some of the challenges in natural language processing.

The next chapter "Basics" describes prerequisites from natural language processing and computational intelligence.

The third chapter "Conception" outlines a first subdivision of the problem, postulates main components of the system and justifies their need. An overview of computational methods and algorithms is given, with the goal of defining a high-level architecture.

Chapter four ("Preprocessing") gives an overview about the tokenization process, describes the entire process of retrieving information on a word base and finally consolidates first atoms to higher level structures, i.e. NGRAMs and phrase fragments.

The final step of the preprocessing – that is the entirety of work on a single phrase – is outsourced to a separate chapter "Dependency Grammar Parsing"; due to it's complexity. An advanced neuro-fuzzy approach is presented, that is independent of additional external databases[8].

The kernel of this work is explained in chapter six: "Information Extraction". The joint of the semantics of the preprocessed phrases; to finally synthesize an (estimation of an) ontology $O$.

"Implementation Remarks" gives a more detailed view on the programming aspects. The co-operation of the software modules is enlightened, e.g. via UML-diagrams.

"Evaluation" examines a concrete system run that is based on the domain D := "The Universe". Several statistics show the uncertainty and effectiveness given a concrete text corpus as input.

Last chapter "Conclusion" gives a project recapitulation and evaluates the proposed and implemented methods.

[6] Verification does primarily mean to rate the uncertainty.

[7] Excluding computer generated source files.

[8] Treebanks.

# Chapter 2

# Basics

## 2.1 Knowledge Representation

### 2.1.1 Ontologies

An Ontology is (in computer science) a formal knowledge representation that aims to get a common understanding[1] of a given knowledge domain $D$. With reference to [MV01], an ontology $O$ and can be defined as a quadruple:

$$O = (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \mathcal{A}0) \tag{2.1}$$

The mathematical objects are given as:

- $\mathcal{C} := \{\ c_1,\ c_2,\ \dots\ \} :=$ a set of $\mathcal{C}$oncepts that is represented by a controlled vocabulary.
- $\mathcal{R} := \{\ r_1 := \varphi_1(c_i, c_j),\ \dots\ \} :=$ a set of binary $\mathcal{R}$elations between the concepts of $\mathcal{C}$. $\varphi_k$ is an abbreviation for a name that describes the relation.
- $\mathcal{HC} := \{\ hc_1, hc_2,\ \dots\ \} := \mathcal{H}$ierarchy of $\mathcal{C}$oncepts that encapsulates concepts of $\mathcal{C}$ for that a hyperonymial relationship $< c_i, c_j >$ exists (where $c_1$ is the hypernym of $c_2$). According to [Wor14], a hyperonym is a "super-subordinate relation" and defined as:

  $$\mathit{isKindOf}(c_i, c_j)\ \rightarrow\ c_j \in \mathit{hyperonym}(c_i)\ \Rightarrow < c_i, c_j > \tag{2.2}$$

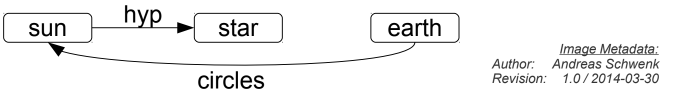
  i.e. we have a hyperonomial relation between the two concepts $c_i$ and $c_j$, in case that $c_i$ is kind of a $c_j$. Then $c_j$ is a hyperonym of $c_i$. We will later abbreviate "hyperonym" with "hyp".
- $\mathcal{A}0 :=$ A set of $\mathcal{A}$xioms that is formulated in a logic language. Axioms control properties of the concepts in $\mathcal{C}$. We do not consider axioms in further considerations ($\mathcal{A}0 := \emptyset$).

An example for a partial ontology of the testing knowledge domain $D :=$ "Universe" could be:

$$O'_{\text{Universe}} = (\ \{\text{sun}, \text{star}, \text{earth}\},\ \{\text{circles}(\text{earth}, \text{sun})\},\ \{< \text{sun}, \text{star} >\},\ \emptyset)$$

Note that we use the symbol $O'$ to indicate an ontology that omits specification of axioms $\mathcal{A}0$. Figure 2.1 depicts a semantic network[2] for the ontology. The graph is derived as follows: $G = (V, E) := (\mathcal{C}, \mathcal{R} \cup \mathcal{HC})$, with $\mathcal{C}, \mathcal{R}, \mathcal{HC} \in O'$ and the mapping $\mathcal{HC}\ \mapsto\ E\ :\ e_i := \mathit{hyp}(c_i \in hc, c_j \in hc)$, with $hc \in \mathcal{HC},\ e_i \in E$ and $c_i \in C$.

Figure 2.1: Semantic Network for $O'_{\text{Universe}}$



[1] Common understanding := shared understanding.

[2] E.g. [SS92] gives an introduction to semantic networks.

## 2.2 Computer Linguistics and Natural Language Processing

As indicated in the introduction, Natural Language Processing (=: NLP) is a multidisciplinary field that unifies knowledge from Computer Science – especially Artificial Intelligence (=: AI) – and Linguistics. NLP is substantial for the definition of interfaces for the interaction between humans and computers. The next sections introduce some subtopics from Linguistics.

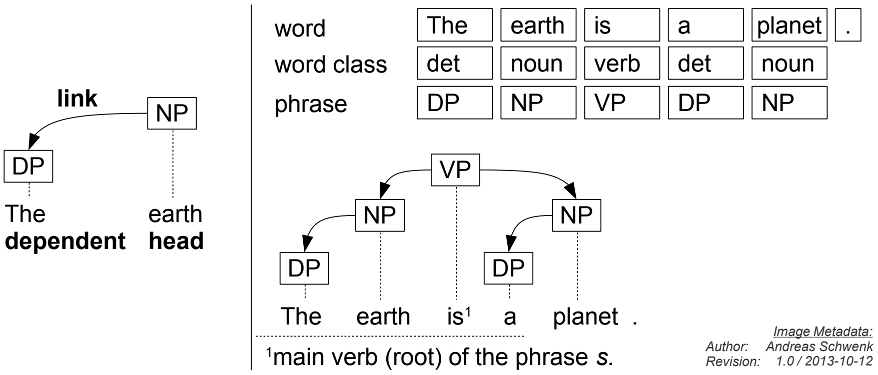
### 2.2.1 Grammar

A phrase is a sequence of words, represented in a certain word order, that is (besides linguistics theory) *implicitly* applied by humans. If this combinatorial order is changed, either the syntax or the semantics, (or both) would be influenced or falsified. The underlying rules are called *grammar*. We can either *generate* (synthesize) language with respect to a given grammar, or *parse* a given phrase; that is the inverse process, i.e. to find the order of applications of grammar rules.

There are two main approaches to categorize the type of grammar. First the *Constituency Grammar*, also called *Phrase Structure Grammar* and second the *Dependency Grammar*. The former was developed by Noam Chomsky[3] and decomposes a phrase step wise into smaller fragments. The dependency grammar was developed by Lucien Tenière. Both types of grammar are strongly equivalent, as described in [Gai65].

As in a later chapter explained, this works mainly focuses on the Dependency Grammar (=: DG) to extract relational information. Figure 2.2 shows a concrete example to illuminate the procedure:

Figure 2.2: Example for an Annotated Dependency Grammar Phrase



The number of nodes is equal to the number of words (in contrast to Phrase Structure Grammar). If two given words are linked, the source is called the *head* and the destination is called the *dependent*. A characteristic of dependency grammar is the significance of the main verb. [CFL12].

We fill focus on the non-trivial parsing process in much detail in a later chapter. The approach is non-standard with respect to literature, but delivers results that are reliable enough to support the process of finding relations between concepts of the ontology.

## 2.3 Computational Intelligence

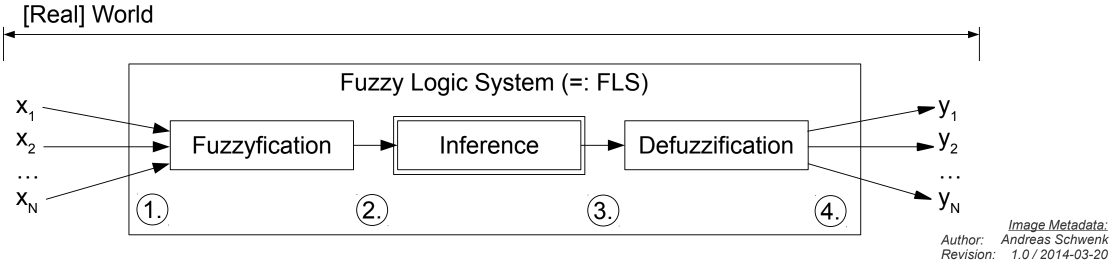
Computational Intelligence (=: CI) is a subset of the Artificial Intelligence (=: AI) and provides algorithms that are derived from biology. To construct the dependency graph a symbiosis of *Fuzzy Logic* and *Artificial Neural Networks* is used; thus we introduce both approaches in subsequent chapters. A reference in literature is e.g. [RN10]. Also [Bar13] was used as a source.

[3] Chomsky found a formalization for languages, and thus created a link between the disciplines computer science and linguistics

### 2.3.1 Fuzzy Logic

A *Fuzzy Logic System* (=: FLS) gives the ability to describe a system's behavior in (formalized) natural language. Thus, the statements – more precise: the set of rules – are rather imprecise ("fuzzy"). Despite the vague formulations, there is a connection to spoken language, that approves validity and usefulness in every day's live. Figure 2.3 shows a generic FLS:

Figure 2.3: Fuzzy Logic System – Overview



The input set $\vec{X}$ of real world values (e.g. measurements) is transformed in the *fuzzyfication*-stage into a degree of membership $\in [0, 1]$. This is performed for every *Linguistic Term* (=: LT) of every *Linguistic Variable* (=: LV)[4].

The next step – *inference* – aggregates the membership values of all linguistic terms for each linguistic variable into a *degree of truth* (=: DoT). This is controlled by rules $R$ in a formalized natural language form; similar to an implication:

$$R_i := \text{IF <premise } p\text{> THEN <conclusion } c\text{>.} \tag{2.3}$$

For example: "IF $\underbrace{\text{weather IS sunny}}_{p_1}$ AND $\underbrace{\text{forecast IS NOT(rain)}}_{p_2}$ THEN $\underbrace{\text{mood IS good}}_{c_1}$".

The aggregation for the set of premises $p = \{p_1, p_2, \dots\}$ may e.g. be performed by a Min/Max Aggregation. Then the Degree of Truth (DoT) for $c_1$ would be $\min\{\mu_1(x), \mu_2(x)\}$ for the logical 'and' operation (or $\max\{\mu_1(x), \mu_2(x)\}$ for the logical 'or') and the *activation* sets the DoT for each conclusion of each rule. Since the number of rules will mostly be $\geq 1$, a subsequent *accumulation*, i.e. a combination of all activations of all rules must be calculated.

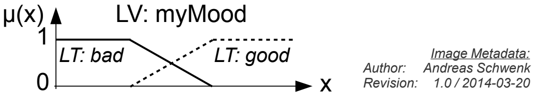
Finally, the *defuzzification* is the inverse part of the fuzzyfication. It is done for each accumulation result of the inference. The generated output $Y$ is back in the (physical) domain of the real world.

The advantage of fuzzy logic is the ability to represent unsharp knowledge in a (natural) form that humans can understand. The membership functions and rules for a FLS may also be formulated and / or adjusted by non-computer-scientists.

### 2.3.2 Artificial Neural Networks

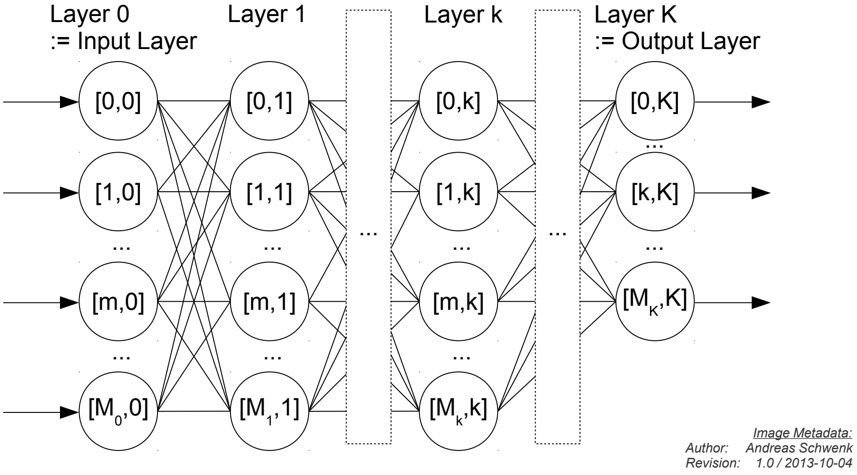
Artificial Neural Networks (=: ANN) are derived from the biological information storage process, i.e. is an abstraction of the (human) brain. A set of interconnected *neurons*, that each has some storage elements (the *weights*), may model any non-linear function behavior. The complexity is dependent on the number of neurons, that itself has mostly to be found empirically. Figure 2.4

[4] A *Linguistic Variable* (=: LV) is a physical quantity. For example the LG "myMood" can be described with the two *Linguistic Terms* (=: LT) "good" and "bad". We may define the following *membership functions* $\mu(x)$ for the representation:



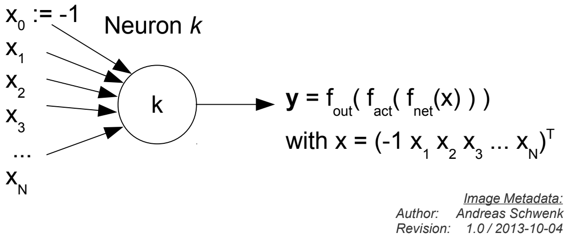
shows a *Multi Layer Perceptron (=: MLP)* that is formally structured[5] in layers and feed-forward connections between each layer. The input and output layers are connected with the real world; i.e. the embedding environment.

Figure 2.4: Multi Layer Perceptron (=: MLP)



Each single artificial neuron [*row*, *column*] is further abstracted by nested functions:

Figure 2.5: A Single Neuron


x0 := -1
x1
x2
x3
...
xN
Neuron k
k
y = fout( fact( fnet(x) ) )
with x = (-1 x1 x2 x3 ... xN)T
Image Metadata:
Author: Andreas Schwenk
Revision: 1.0 / 2013-10-04

For applications in this project we will always use *weighted sum* for the net-function, *sigmoid* for the activation-function and *identity* for the output function[6]:

$$f_{net}(X) := \sum_{n=0}^{N} x_i \cdot w_i \tag{2.4}$$

$$f_{activation} := \frac{1}{1 + e^{-f_{net}}} \tag{2.5}$$

$$y := f_{output} := f_{activation} \tag{2.6}$$

An artificial neural network must be trained[7] to get some valid output for a given input. This can be done either manually or automatically. Training is the process that defines the weights by numbers $\in \mathbb{R}$. A widely used algorithm for automatic learning is the *Backpropagation Learning Algorithm*; developed for Multi-Layer Perceptron networks (compare to figure 2.4) by David E. Rumelhart [RHW86]. Such a training is done by given *Training Set* (=: *TS*). A training set is defined as a tuple *TS* = (*InputValues*, *OutputValues*). Each training set is iteratively applied in a given number of *epochs*[8] to the system in- and output. The algorithm adjusts the weights until the

[5] Note that the interconnections of human brain have not such a feed-forward structure, but the abstraction here simplifies further considerations. E.g. the back-propagation learning algorithm relies on a regular structure.

[6] See literature for more detailed explanations.

[7] The training of artificial neural networks is called *supervised learning*.

[8] The number of epochs within back propagation learning must be very high, and consequently the computing

remaining error is below a given threshold, i.e. the output is calculated by a given training set's input and is then compared to the estimated value.

#### 2.3.2.1 Example

The following example anticipates an application of the real implementation in Java (developed for this project) that will be presented in a later chapter. In this case, the logical AND-function is applied to a MLP-network[9].

Figure 2.6: Error Development as a Function of the Current Epoche

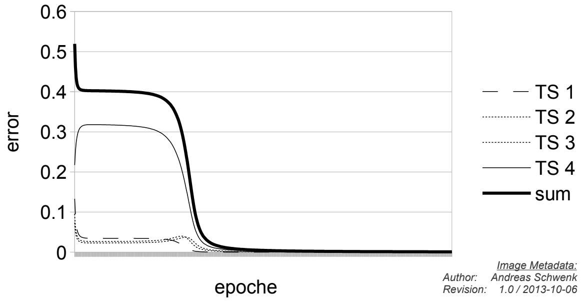


Image Metadata:
Author: Andreas Schwenk
Revision: 1.0 / 2013-10-06

Table 2.1: ANN-Example: Real output

| idx. | input 1 | input 2 | expected | real output | Δ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0.0051 | 0.0051 |
| 1 | 0 | 1 | 0 | 0.0177 | 0.0177 |
| 2 | 1 | 0 | 0 | 0.0206 | 0.0206 |
| 3 | 1 | 1 | 1 | 0.9666 | 0.0334 |

Figure 2.7: Weights for the "AND-test"($\eta = 2.0$, 150 epochs, $\alpha = 0.5$)

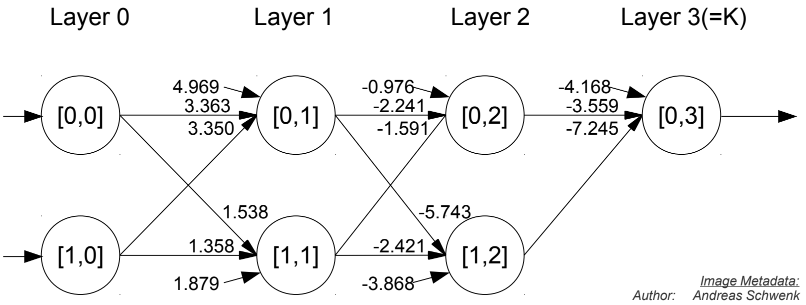


Image Metadata:
Author: Andreas Schwenk
Revision: 1.0 / 2013-10-06

In the project we use Artificial Neural Networks to represent uncertainty that is inherently given, as the user trains the system for being able to do Dependency Grammar Parsing.

power must be taken into account.

[9] The number of layers and the number of neurons per layer was found empirically. The ideal dimensionality is an multi-criteria optimization problem with the objectives of a minimized number of needed epochs until convergence is sufficient and simultaneously a minimized number of neurons. Furthermore, the applied internal parameters are set to: Learning rate $\eta = 2.0$; momentum $\alpha = 0.5$; 500 epochs.

## 2.4 Mathematical Notation

Some notations were especially defined for this work, and are briefly summarized in the following. All other notation is derived from mathematics, theoretical computer science, as well as graph theory. Refer e.g. to [EP08].

### 2.4.1 Phrases

A phrase $p$ is represented as a (hierarchical) linked list, consisting of words $w$:

$$\begin{aligned} p &:= \vdash w_1 \rightarrow w_2 \rightarrow \dots \rightarrow w_{|p|} \dashv \\ p &:= w_1 w_2 \dots w_{|p|}. \end{aligned} \tag{2.7}$$

with $|p|$ the length of the phrase, i.e. the number of its words. We also use $|w|$ for the same purpose. Any sequence of words in $p$ can be declared as a fragment $f$. We restrict the set of fragments $F$ to { NP := Nominal Phrase, VP := Verbal Phrase }. If the fragment begins starting from word $w_i$ and ends at word $w_i + k$ (included), we denote for a Nominal Phrase:

$$\begin{aligned} p &:= \vdash w_1 \rightarrow w_2 \rightarrow \dots \rightarrow \mathrm{NP}(w_i \rightarrow w_{i+1} \rightarrow \dots \rightarrow w_{i+k})) \dots \rightarrow w_{|p|} \dashv \\ p &:= w_1 w_2 \dots \mathrm{NP}(w_i w_{i+1} \dots w_{i+k})) \dots w_{|p|}. \end{aligned} \tag{2.8}$$

A property of a word $w$ is expressed by a function on the word. E.g. the word-class of a word $w$ is represented as:

$$\mathit{class}(w) \in \{\mathrm{Noun}, \mathrm{Verb}, \mathrm{Adjective}, \dots\} \tag{2.9}$$

### 2.4.2 Grammar

A Grammar is represented by a graph $G = (V, E)$, i.e. consists of a set of vertices (nodes) $V = \{v_1, v_2, \dots, v_n\}$ and a set of edges $E = \{e_1, e_2, \dots, e_m\}$. $G$ is a tree and therefore the number of cycles is zero. For the number of vertices $n$ and the number of edges $m$ we have the constraint:

$$m = n + 1 \tag{2.10}$$

$G$ is directed and for a Dependency Grammar, each edge $e \in E$ is given as:

$$e = (v_1, v_2) \in E = (h, d) \tag{2.11}$$

with $h :=$ head and $d :=$ dependent. The root node is characterized as: $\exists_{e \in E} \;:\; e = (h, d) \wedge \mathit{eq}(h, \emptyset)$; with the predicate $\mathit{eq}(x, y) := \mathit{equals}(x, y) := x = y$. In case $G$ is a Dependency Grammar[10] the number of words $|w| := |p|$ is equal to the number of vertices $n := |V|$, i.e. each word $w_i$ is a vertex $e_i$.

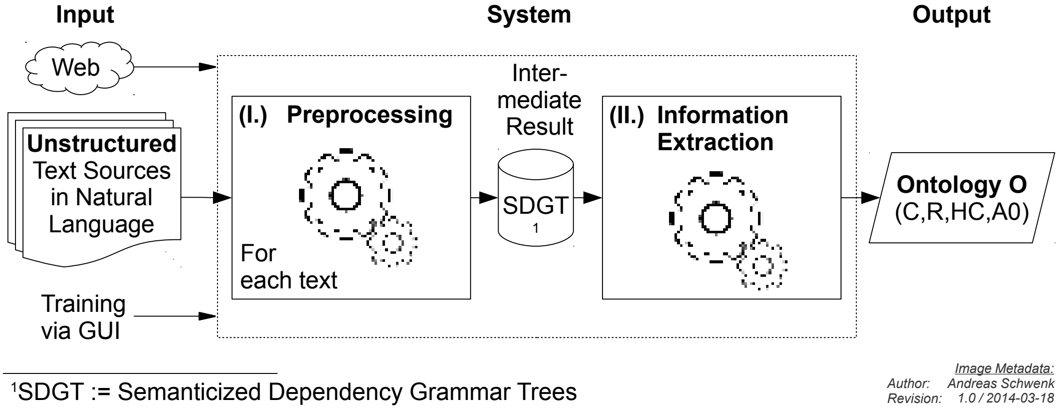
[10] Phrase Structure Grammars are not considered in this work.

# Chapter 3

# Conception

A first subdivision of the entire system is to separate *preprocessing* and *information extraction* in form of a pipeline structure. For former will gather unconsolidated data for each given phrase of the input corpus, while the latter will build an ontology $O$.

Figure 3.1: Overview of the System



We define the system boundaries, i.e. the set of inputs and outputs as depicted in figure 3.1. Obviously, a set of unstructured English language texts is applied to the system. To enrich[1] each processed word with semantics, the preprocessing subcomponent retrieves information from the web, e.g. Wiktionary and other sources. Some internal steps apply implicit rules that require some manual supervised learning, so a *training* via the Graphical User Interface (=: GUI) can be performed. Note that this only has to be done once, and may further be done subsequently to decrease uncertainty, if necessary. The *intermediate result*, i.e. the output of the preprocessing system, is represented as an extended form of a dependency grammar:

$$\text{IntermediateResult} \;:=\; \cup_{\text{Phrases}} : \{\ \text{DependencyGrammar} \cup \text{Semantics} \cup \text{FurtherInformation}\ \}$$

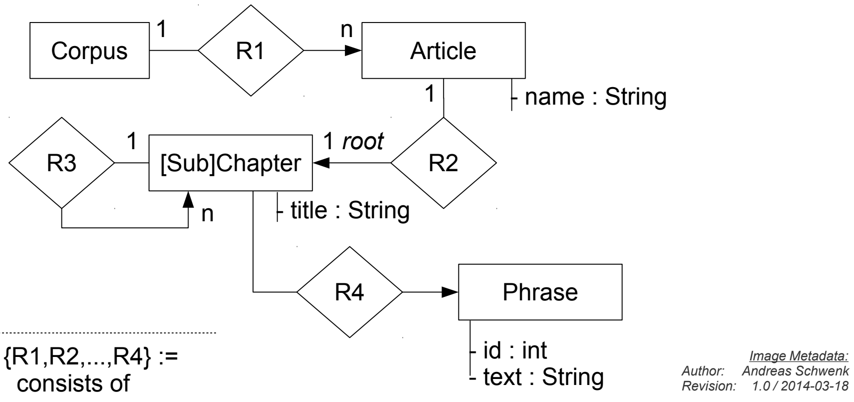
The next sections describe briefly the internal steps for the two main subcomponents. Complex tasks are expounded in separate chapters.

[1] Annotate words with information.

## 3.1 Corpus Control

The corpus is applied via a set of local stored text files[2] to the system. Each of the files may have a hierarchical (tree-based) structure with an arbitrary depth[3]. The Entity Relationship Diagram (=: ERD) depicted in figure 3.2 gives an overview of the construction. The first part "Preprocessing" works on a phrase-base, i.e. each phrase is analyzed separately; while the second part "information extraction" requires the analyzed data of the whole corpus.

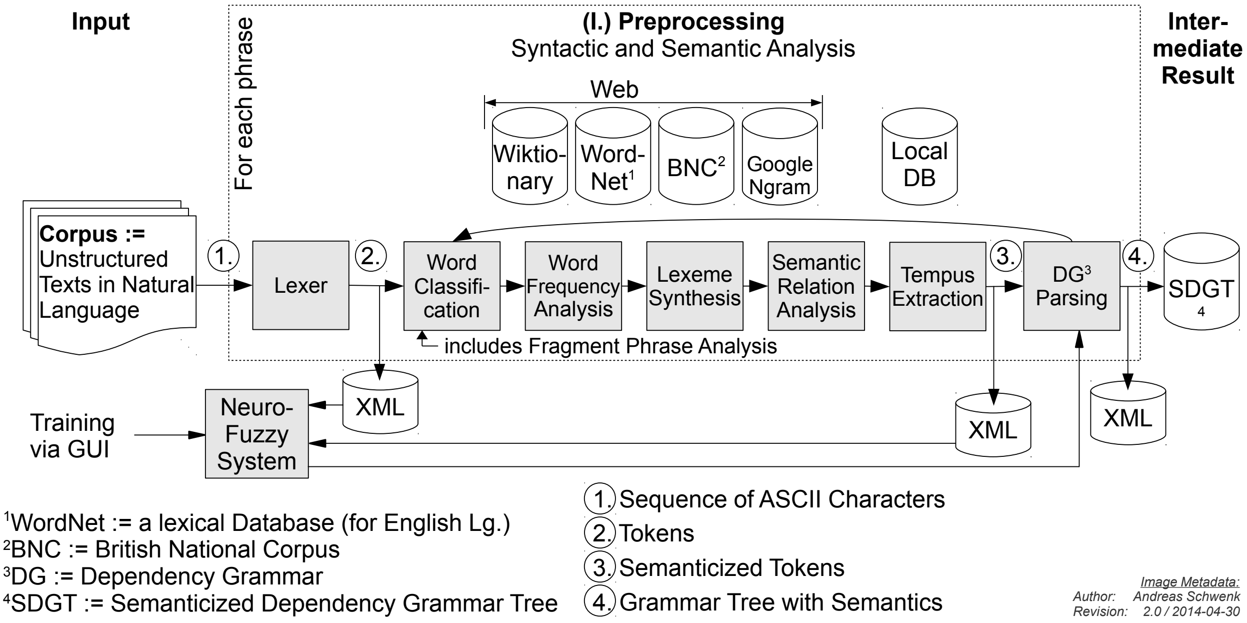
Figure 3.2: Entity Relationship Diagram: Corpus Structure



## 3.2 Preprocessing

"Preprocessing" is subdivided into a set of stages. Figure 3.3 shows a chain of functional blocks. These mostly rely on previous steps and each enlarges the underlying data structures.

Figure 3.3: Overview of the Preprocessing Part


Input
For each phrase
(I.) Preprocessing
Syntactic and Semantic Analysis
Inter-mediate Result
Web
Wiktio-nary
Word-Net¹
BNC²
Google Ngram
Local DB
Corpus := Unstructured Texts in Natural Language
1.
Lexer
2.
Word Classifi-cation
Word Frequency Analysis
Lexeme Synthesis
Semantic Relation Analysis
Tempus Extraction
3.
DG³ Parsing
4.
SDGT ⁴
includes Fragment Phrase Analysis
XML
XML
XML
Training via GUI
Neuro-Fuzzy System
¹WordNet := a lexical Database (for English Lg.)
²BNC := British National Corpus
³DG := Dependency Grammar
⁴SDGT := Semanticized Dependency Grammar Tree
1. Sequence of ASCII Characters
2. Tokens
3. Semanticized Tokens
4. Grammar Tree with Semantics
Image Metadata:
Author: Andreas Schwenk
Revision: 2.0 / 2014-04-30

[2] Format: ASCII-Code / UTF8.

[3] The structure is realized as follows: (1.) Each chapter starts with a headline that is introduced with a number of repeating asterisks "∗"; the number represents the depth, i.e. "*" introduces the entire article; "**" is on a chapter base; "***" is the depth of a subchapter etc. (2.) The text of the current chapter begins in the following line and ends right before the next headline.

For run time optimization, i.e. faster access to web sources, as well as storing already analyzed data, intermediate results – denoted as ⓘ in picture 3.3 – are persistently output to XML-files. Each data set encloses its predecessor ($1 \subseteq 2 \subseteq 3 \subseteq 4$) and may be briefly summarized as:

1. Sequence of ASCII Characters := Contains the text of a phrase on a character base.
2. Tokens := Word segmented list, consisting of words, numbers and punctuation characters.
3. Semanticized Tokens := Annotated words.
4. Grammar Tree with Semantics := Hierarchical grammar representation: (1.) Grammar on a Fragment Phrase base (Nominal Phrases, Verbal Phrases) (2.) Grammar on a word base.

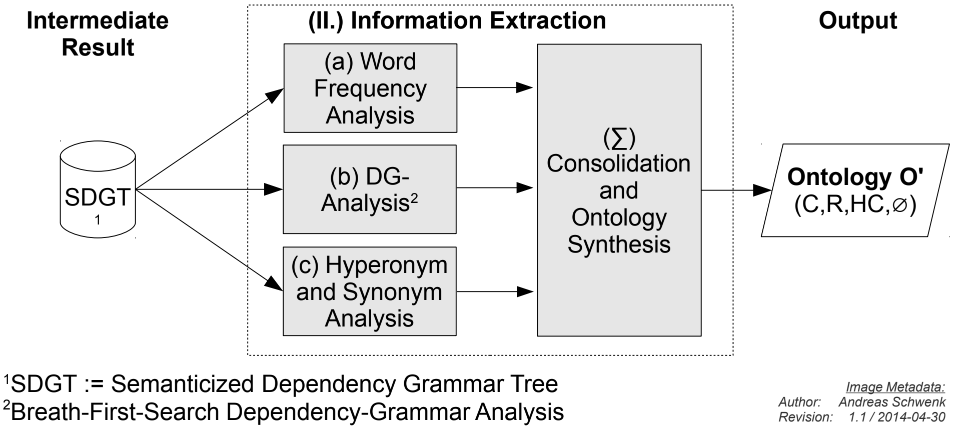
In summary, preprocessing does tokenization, word classification including fragment phrase analysis, lexeme synthesis, semantic relation analysis, word frequency analysis and meaning gathering on a word base; while tempus detection operates on a phrase base. The order must be kept for most blocks, since e.g. tempus extraction relies on word-classes etc. Chapter 4 describes most of the preprocessing steps in detail. Since dependency grammar parsing is a non-trivial task, the approach is described in separate chapter 5.

## 3.3 Information Extraction

The "Information Extraction" part uses the preprocessed data in form of a hierarchical Semanticized Dependency Grammar Tree (=: SDGT) for each of the phrases of the corpus. SDGT is a tree-based data structure that annotates all the gathered data to the underlying words resp. directly to the phrase.

An overview is given in figure 3.4. Subroutines (a) – (c) build a first estimation of the output ontology $O'$, while ($\sum$) performs some post processing work, that e.g. unifies partial ontolgies from the former steps and synthesis the output data in the Web Ontology Language (=: OWL) format. The Word Frequency Analysis estimates the concepts $\mathcal{C}$; Dependency Grammar Analysis estimates the relations $\mathcal{R}$; and the Hyeronym and Synonym Analysis estimates the Hierarchy of Concepts $\mathcal{HC}$.

Figure 3.4: Overview of the Information Extraction Part


Intermediate Result
(II.) Information Extraction
Output
(a) Word Frequency Analysis
(b) DG-Analysis²
(c) Hyperonym and Synonym Analysis
SDGT¹
(∑) Consolidation and Ontology Synthesis
Ontology O'
(C,R,HC,∅)
¹SDGT := Semanticized Dependency Grammar Tree
²Breath-First-Search Dependency-Grammar Analysis
Image Metadata:
Author: Andreas Schwenk
Revision: 1.1 / 2014-04-30

Refer to chapter 6 for detailed information on algorithms and data structures.

## 3.4 Graphical User Interface Design

User interaction and a content oriented presentation of information is realized with a custom Graphical User Interface (=: GUI) – implemented in Java. While plenty of GUI-APIs exist, this custom version is designed with a specialized focus on Natural Language Processing. This way, most of the data is shown in form of lists and trees; each consisting of generic elements.

Each of these elements has the ability to display a text or a diagram, e.g. a word or probability distributions. Visual scalability allows to observe certain details.

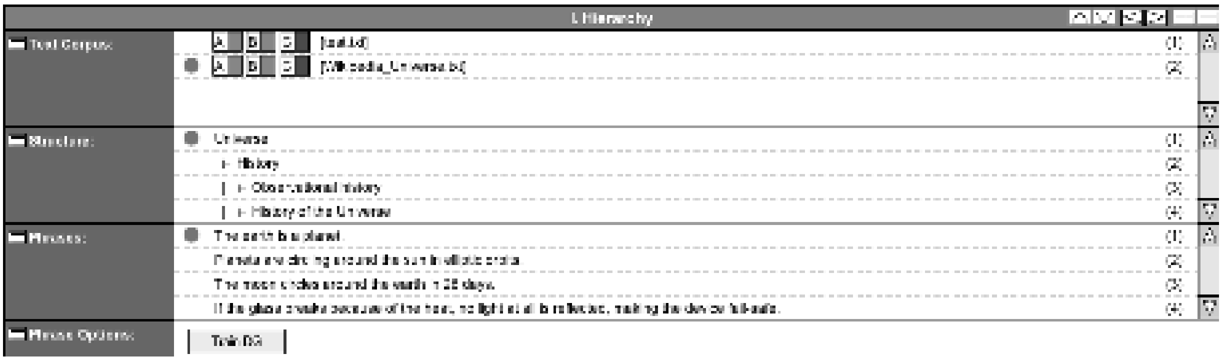
### 3.4.1 Corpus Management

Figure 3.5 displays the part of Corpus Management: All texts of the corpus, that are provided in the UTF8-format and are placed in the input-directory on the disk, are listed. Each can individually be processed to one of the states $\{A, B, C\}$:

- $A$ performs word segmentation and synthesizes an XML-file containing tokens.
- $B$ semanticizes all tokens that are of type *word*. Semanticization annotates all preprocessing information to the word. Afterwards the above phrase-based processings are performed.
- $C$ parses the dependency grammar.

Since preprocessing examines phrases, the hierarchical text structure can be browsed; and results in a list of phrases. Selection of a single phrase switches the panel "phrase analysis" (see below).

Figure 3.5: Graphical User Interface: Hierarchy



The earth is a planet.

Planets are circling around the sun in elliptic orbits.

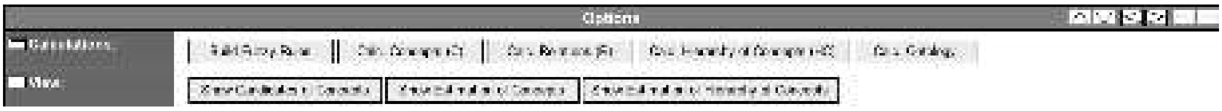
The moon circles around the earth in 28 days.

If the glass breaks because of the heat, no light at all is reflected, making the device fail-safe.

### 3.4.2 Options

The most options in the panel in figure 3.6 are ontology referred and divided into calculation and view actions. "Build Fuzzy Rules" is related to the Dependency Grammar Learning process. The other choices start synthesis or show the parts of the ontology.

Figure 3.6: Graphical User Interface: Options



Options
Calculations:
Build Fuzzy Rules
Calc. Concepts (C)
Calc. Relations (R)
Calc. Hierarchy of Concepts (HC)
Calc. Ontology
View:
Show Cardinality of Concepts
Show Similarity of Concepts
Show Similarity of Hierarchy of Concepts

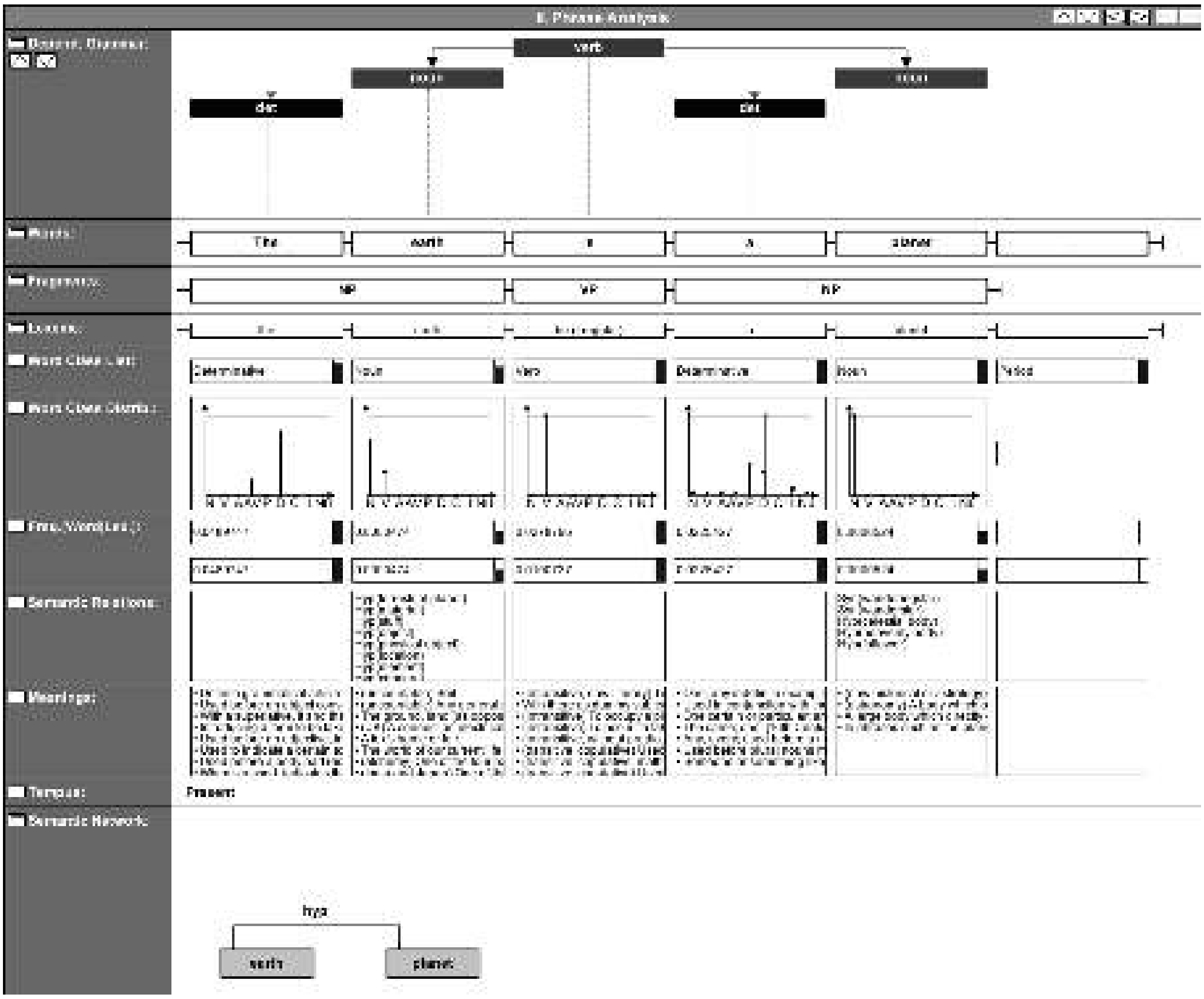
### 3.4.3 Phrase Analysis

Figure 3.7 shows the entirety of information for the currently selected phrase. Details are explained in later chapters. A brief overview is given in table 3.1:

Table 3.1: Phrase Analysis GUI

| Part | Description |
|---|---|
| Dependency Grammar | Only visible, if preprocessing state is $C$. |
| Words | Tokenized phrase. |
| Fragments | Subphrases $\in$ {NP := Nominal Phrase, VP : Verbal Phrase}. |
| Lexeme | Generic form of the current word. |
| Word Class List | Estimated word class $\in$ { Noun, Verb, Adjective . . . }. |
| Word Class Distribution | Details for word class estimation: black colored values := Wiktionary estimation; red colored values := fragment analysis corrections. |
| Frequency (Word, Lexeme) | Probablity that { (a) the word and (b) the lexeme } occurs in the language statistically. |
| Semantic Relations | Lists synonyms and hyperonyms for the current word. |
| Meanings | Lists a set of meanings for the current word. |
| Extraction | Information extraction. Gives first estimations for the ontology in form of a semantic network. |

Figure 3.7: Graphical User Interface: Phrase Analysis



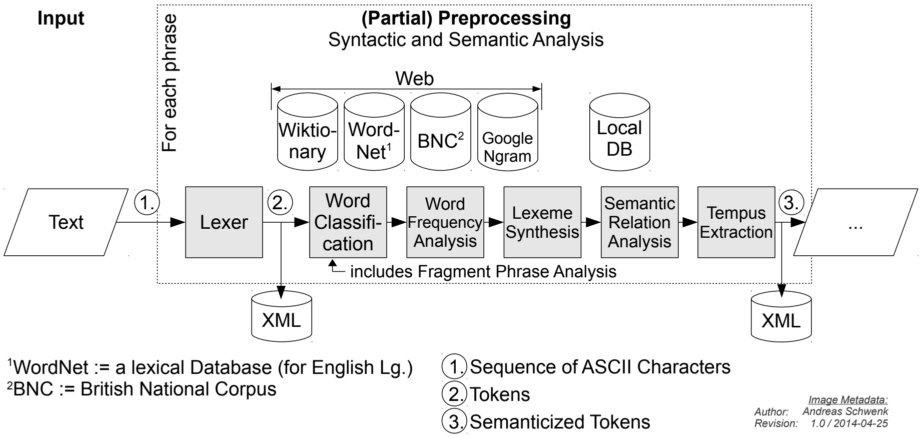
This exposition is incomplete. Further windows and forms, will be presented in the appropriate subchapters; e.g. windows related to ontology synthesis.

# Chapter 4

# Preprocessing

Preprocessing comprises a Semantic and Syntactic Analysis of each a phrase $p$ separately. This chapter focuses on all prepecessing parts, except Dependency Grammar Parsing that will be discussed in a later chapter, to keep comprehensibility. Refer to figure 4.1 to get an overview of the partial implementation.

Figure 4.1: Partial Preprocessing Steps



## 4.1 Lexical Analysis

Tokenization transforms the character-stream into a list of words. Equation 4.1 shows a Type–3 Grammar $G(\mathcal{L}_3)$ (:= Regular Grammar) in EBNF[1]. We define a word $w_i$ := <Token>$_{EBNF}$ and parse deterministically according to the rules. [Wir96] is a classical source for compiler construction, and also describes the lexical analysis.

$$\begin{array}{lll}
\text{<UppercaseLetter>} & ::= & \text{``A'' | ``B'' | ``C'' | \ldots | ``Z''.} \\
\text{<LowercaseLetter>} & ::= & \text{``a'' | ``b'' | ``c'' | \ldots | ``z''.} \\
\text{<Digit>} & ::= & \text{``1'' | ``2'' | \ldots | ``9''.} \\
\text{<Digit0>} & ::= & \text{``0'' | <Digit>.} \\
\text{<Word>} & ::= & \text{[ <UppercaseLetter> ] \{ <LowercaseLetter> \}.} \\
\text{<Number>} & ::= & \text{<Digit> \{ <Digit0> \}.} \\
\text{<PunctuationCharacter>} & ::= & \text{``.'' | ``,'' | ``;'' | ``\textvisiblespace''.} \\
\text{<Token>} & ::= & \text{<Word> | <Number> | <PunctuationChar>.} \\
\text{<Text>} & ::= & \text{\{ <Token> \}.}
\end{array} \tag{4.1}$$

[1] EBNF := Extended Backus–Naur Form

We finally get a phrase $p := \{w_1, \dots, w_{|p|}\}$ that consists of a list of tokens[2]. Each subsequent consideration in the following processing steps will be at a word-based granularity at minimum.

### 4.1.1 N-Grams

A post-processing step (with respect to Lexical Analysis) is a $n$-gram determination, i.e. a concatenation of $n$ tokens to a contiguous sequence. We only consider 2-grams to e.g. combine the words “because of” or “light year”. Implementation iterates over each word $w_i$ and checks, if there exists a Wiktionary entry (more details on Wiktionary are given in the next section):

$$\exists_{w_k \in \mathit{Wiktionary}} \; : \; \mathit{equal}(w_k, \mathit{concatenate}(w_i, w_{i+1})) \;\rightarrow\; \mathit{is2gram}(w_k). \tag{4.2}$$

The *Google Ngram-Viewer*[3] could be used for further observations of $n$-grams, but is not considered here.

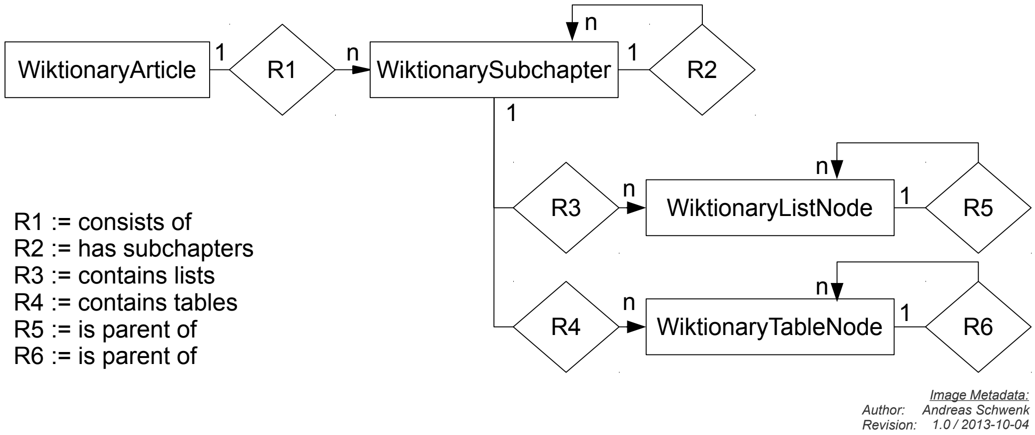
## 4.2 Syntactic and Semantic Analysis

For syntactic and semantic analysis we will consecutively retrieve information from several web-sources. For a first estimation we restrict the focus on *Wiktionary*; a freely available content dictionary. The structure for a word $w$ provides (at least) the following data[4]:

$$\begin{array}{lcl}
\text{<Meaning>} & := & \text{Text.} \\
\text{<Synonym>} & := & \text{Text.} \\
\text{<WordClass>} & := & \text{“Noun|Verb|\dots”} \; \{ \text{ <Meaning> } \} \; \{ \text{ <Synonym> } \}. \\
\text{<English>} & := & \text{<Etymology> <Pronunciation>} \; \{ \text{ <WordClass> } \} \\
 & & \text{<UsageNotes> <RelatedItems> .} \\
\text{<Language>} & := & \text{<English>} \; | \; \dots . \\
\text{<WikiEntry>} & := & \{ \text{ <Language> } \}.
\end{array} \tag{4.3}$$

The base entry <WikiEntry> itself is a HTML-file[5]. Each file is parsed via a DOM-parser[6] (refer to [DOM]) and assembled into a hierarchical data structure, depicted in figure 4.2. We obtain the needed information as shown in listing 4.1. This process is based on the assumption that each Wiktionary Article has the following generic and hierarchically structure:

Figure 4.2: Wiktionary Entity Relationship Diagram (=: ERD)



[2] We will call tokens (despite its original sense) as words in the next chapters; even if it is a punctuation mark etc.

[3] `https://books.google.com/ngrams`

[4] The website for a word $w$ can be found via the concatenation `http://en.wiktionary.org/wiki/` $\circ$ $\boxed{\text{word}}$.

[5] HTML := Hypertext Markup Language

[6] DOM := Document Object Model

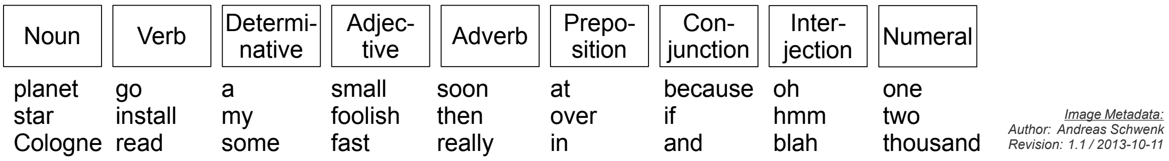
Listing 4.1: Wiktionary Parsing

```
program ExtractAnnotations(word w)
  Article a = preparse(WiktionaryArticle(w))
  for all Subchapters s of a do
    if headline(s) equals "English" then
      for all following subordinated Subchapters s2 of s do
        if headline(s2) equals ("Noun"|"Verb"|...) then
          extract each child-node n from List 0 of s2
          (n each contains a unique meaning m)
        else if headline(s2) equals "Synonyms" then
          extract each child-node n from List 0 of s2
          (n each contains synonym s)
        end if
      end for
    end if
  end for
end program ExtractSemantics
```

In the following, we call an attributed word, i.e. a word $w$ of a phrase $p$ with additional information that extend $w$, a *Semanticized Token* (=: STk). Examples for attributes are e.g. the wordclass $class(w)$ or the lexeme $lex(w)$.

### 4.2.1 Word Classification

Figure 4.3: English Word Classes



| Noun | Verb | Determi-native | Adjec-tive | Adverb | Prepo-sition | Con-junction | Inter-jection | Numeral |
|---|---|---|---|---|---|---|---|---|
| planet | go | a | small | soon | at | because | oh | one |
| star | install | my | foolish | then | over | if | hmm | two |
| Cologne | read | some | fast | really | in | and | blah | thousand |

Image Metadata:
Author: Andreas Schwenk
Revision: 1.1 / 2013-10-11

The set of English word classes (also called "type of speech") with examples for each class are depicted in figure 4.3. In some cases we can determine the word class for a word $w_i$ by a simple lookup to get an contravalent result:

$$class(w_i) := \text{Noun} \oplus \text{Verb} \oplus \ldots \tag{4.4}$$

Other words belong to more than one word class and are thus *homonyms*[7]. Determination of class($w_i$) can not only rely on $w_i$, since the size of the resulting set

$$class(w_i) := \{wc_1, wc_2, \ldots, wc_N\}, \quad wc_i \in \{\text{Noun, Verb, Adjective, } \ldots\} \tag{4.5}$$

is greater than one (wc := word class), if $eq(isHomonym(w), true)$.

#### 4.2.1.1 First Estimation

A first approximation to solve homonymy is to involve the number of meanings for a word class $wc_i$ of a word $w_i$ given by a dictionary (here Wiktionary). Each wc$_i$ is semantically defined with respect to the word class as:

$$def(wc_i) := \{m_1, m_2, \ldots, m_M\}, \quad M := |def(wc_i)| \tag{4.6}$$

with $m_i$ := the meaning, i.e. a phrase that equals a definition for $w$ with respect to the wordclass $wc_i$. We define the Probability Mass Function (=: PMF) $P_{wc}$ that shows the context-free[8]

[7] Homonyms are words that have different meanings, but the same spelling. *Polysemes* are polysemous homonyms: Their origin is the same [Stu].

[8] Context-free := $w$ is considered isolated.

distribution of the probability of word classes for a word $w_i$:

$$P_{wc}(w_i) := [\ |def(wc_1)|\ \ |def(wc_2)|\ \ \ldots\ \ |def(wc_N)|\ ] \cdot \frac{1}{\sum_i\ |def(wc_i)|} \tag{4.7}$$

The weight $W := 1/\left(\sum_i\ |def(wc_i)\right)|$ forces the sum of all values $|def(wc_i)|$ to be one. In the special case that the word class cannot be determined, i.e. there does no Wiktionary article exist, we assume the word to be a noun; since the probability of being a proper noun is highest:

$$P_{wc}(w_i) := [2\ 1\ 1\ \ldots] \cdot W \tag{4.8}$$

Thus, $p(\text{Noun})$ has the highest value, but is set lower to one, since there is uncertainty.

**Example:** For the word “earth” we retrieve from Wiktionary[9]:

$$\begin{aligned} P_{wc}(\text{earth}) \ &:= \ [\ |def(\text{Noun} := \{\text{ProperNoun} \cup \text{Noun}\})|\ \ |def(\text{Verb})|\ ] \cdot W \\ &= [10\ \ 4] \cdot \tfrac{1}{10+4} = [0.714\ \ 0.286] \end{aligned} \tag{4.9}$$

The first word class estimation (=: est) is calculated by the frequency of usage[10]:

$$class_{est}(w_i) := \max\{P_{wc}(w_i)\}\ (=:\ ewc := \text{estimated word class}) \tag{4.10}$$

Thus without considering the context, we estimate $class(\text{earth}) := \text{Noun}$ and attribute the probability to be

$$p(class(\text{earth}))\ :=\ 0.714 \mathrel{\hat{=}} 71.4\% \tag{4.11}$$

#### 4.2.1.2 Refinement by Crisp Logic

The uncertainty of the word class of word $w_i$ may be reduced, if the context is considered and may change the word class candidate that was estimated above. We restrict the context to a subphrase (or *fragment*) $f$ that is defined by the surrounding words of $w_i$:

$$\begin{aligned} f \ &:= \ [\ w_{i-k}\ \ w_{i-k+1}\ \ \ldots\ \ w_{i-1}\ \ \boxed{w_i}\ \ w_{i+1}\ \ \ldots\ \ w_{i+l-1}\ \ w_{i+l}\ ]\ \ \subseteq\ p \\ &:= \ [f_0\ \ f_1\ \ \ldots\ \ f_{|f|}] \end{aligned} \tag{4.12}$$

The word class of $w_i$ can be interpreted as a crisp logic function[11] of constraints, based on surrounding words $w_{i+j},\ j \in \mathbb{Z}$:

$$class(w_i) := \begin{cases} wc'_i & \text{if}\ \ \forall_{j\in\mathbb{Z}}\ :\ eq(\ \varphi_j(w_{i+j}),\ \phi_j\ ) \\ wc_i & \text{otherwise} \end{cases} \tag{4.13}$$

$wc'_i$ is the word class that is set, if the crisp logic function is true. $\varphi_j$ is a function of a the word $w_{i+j}$ and gets a property. The set of properties is listed in table 4.1. $\phi_j$ is a constant that must match with the result of $\varphi(w)$. Matching for a single word $w_{i+j}$ is tested by the predicate $eq(x, y) := equals(x, y) := x = y$.

[9] def(ProperNoun) := { Our planet – third out from the Sun }. def(Noun) := { Soil, rock-based material, ground, connection electrically to the earth, fox’s home of lair, world of our current life, one of the four basic elements, (India and Japan) one of the five basic elements, (Taoism) one of the five basic elements }. def(Verb) := { to connect electrically to the earth, to bury, to hide, to burrow }.

[10] We assume that the number of meanings for each word class correlates with the distribution in natural language texts and thus, this strategy implies high uncertainties.

[11] Crisp Logic := Classic Logic.

Table 4.1: Functions on a Word $w$

| $\varphi(w)$ | Description |
|---|---|
| $lex(w)$ | Lexeme of word $w$. |
| $lexType(w)$ | Type of the lexeme of word $w$. |
| $ewc(w)$ | Estimated Word Class of word $w$ that is determined by equation 4.10. |

Listing 4.2 gives an example for a set of rules that are syntactically formulated in a custom script language[12], to handle the *pattern matching*. Each condition on the left-hand side represents a fragment $f$ that aims to match at arbitrary positions within a phrase $p$. The right-hand side defines the word class *for each* of the words in the matched fragment; and thus is an extension to equation 4.13, that itself determined only the word class for a *single* word. The entire fragment $f$ is moreover tagged to a type $\in \{\mathrm{NP}, \mathrm{VP}\}$, i.e. this script of crisp rules additionally detects Nominal- (NP) and Verbal Phrases (VP). A more detailed view on *Lexemes* is given in the next subchapters.

Listing 4.2: Crisp Rules for Fragment Extraction and Word Class Refinement

```
[LEX=be LEXTYP=PresentParticiple EWC=Adverb] -> VP [Verb Verb Adverb].
[LEX=be LEXTYP=PresentParticiple]            -> VP [Verb Verb].
[LEX=be LEXTYP=PastParticiple]               -> VP [Verb Verb].
[LEX=have LEXTYP=PresentParticiple]          -> VP [Verb Verb].
[LEX=have LEXTYP=PastSimple]                 -> VP [Verb Verb].
[EWC=Determinative EWC=Noun]                 -> NP [Determinative Noun].
[LEX=a EWC=Noun]                             -> NP [Determinative Noun].
[EWC=Adjective EWC=Noun]                     -> NP [Adjective Noun].
[EWC=Verb]                                   -> VP [Verb].
[EWC=Noun]                                   -> NP [Noun].
```

The script can be edited / extended via an usual text editor and is interpreted at run-time. A regular language ($\mathcal{L}_3$) for parsing is sufficient and was implemented deterministically. Each rule is tried to be applied on each possible phrase position in the given rule-order. Rules should be ordered by descending length, i.e. by a decreasing number of words, since short rules would otherwise hide the semantics of more complex rules. The described and implemented approach could indeed be extended to use Fuzzy Logic. This is here omitted due to two reasons:

1. Complexity in implementation and adjustment ("overhead").
2. Comparision of words to given word-constants (e.g. lexemes) can only be done with crisp logic.

### 4.2.2 Lexeme Detection

Words $w_i$ were up to now treated, as parsed by the tokenizer. To distinguish the basic unit of meaning from $w_i$, we define the lexeme $lex(w_i)$, that extracts the generic form, i.e. it removes declination endings from verbs, plural forms from nouns etc. Table 4.2 lists the set of lexemes that the system is capable of extracting:

Table 4.2: Types of Lexemes (=: lextypes)

| Lex.Type | Description |
|---|---|
| Unchanged | Either $w_i$ is not inflected, or the system could not detect that $w$ is a lexeme. |
| Plural | $w_i$ is a plural. |
| PresentParticiple | $class(w_i) :=$ Verb and $w_i$ is a present participle. |
| PastSimple | $class(w_i) :=$ Verb and $w_i$ is of type past simple. |
| PastParticiple | $class(w_i) :=$ Verb and $w_i$ is a past participle. |
| Irregular | $class(w_i) :=$ Verb with irregular inflection. |

A first trivial step removes an apostrophed ending; e.g. "universe's" is replaced by "universe".

[12] Postulated for this project.

**Examination of Nouns:** Nouns are solely investigated toward the plural lexeme type. A first approach could be to remove the ending ‘s’ resp. ‘es’; but is not sufficient for many nouns, especially those with foreign-language origin[13]. A more reliable approach is to use *Wiktionary* as a dictionary. Since semantics – expressed by a set of meanings $def(wc_i)$ – is already extracted for usage in equation 4.7, one may reuse this information. Wiktionary generally includes the term ‘plural from of <$lex(w_i)$>’ within one of the meanings $m_i$. From this starting point, we derive the algorithm to extract the plural-lexeme for a word $w$:

$$\begin{aligned}&\underline{\text{for}}(m_i \in def(w, class(w) := \text{Noun})\ ):\\ &\quad \{\omega\} := \text{words of meaning } m_i;\\ &\quad \exists_{i\in\mathbb{N}_0\ \wedge\ i<|\omega|-3}\ :\ eq(\omega_i, \text{“plural”}) \wedge eq(\omega_{i+1}, \text{“form”}) \wedge eq(\omega_{i+2}, \text{“of”})\\ &\qquad \rightarrow\ lex(w) := \omega_{i+3};\end{aligned} \tag{4.14}$$

with $\omega_i$ a list of words that outline the current meaning $m_i$.

**Examination of Verbs:** Lexeme extraction for verbs is divided into three different approaches:

1. Manual programming.
2. Static irregular verb list.
3. Wiktionary bases extraction.

Manual programming relies on a lookup-table that has (at least) the entries listed in table 4.3

Table 4.3: Manual Lexeme Extraction for Verbs

| $w$ | $lex(w)$ | $lextype(w)$ |
|---|---|---|
| has | have | Irregular |
| had | have | PastParticiple |
| is | be | Irregular |
| as | be | PastSimple |

An irregular verb list has been taken from [Irr14] and transformed to a CSV[14]-file. The layout may be expressed by (Relational Database scheme):

$$(A, dt, I) := (\{\text{Present}, \text{PastSimple}, \text{PastParticiple}\}, \{\text{String}, \text{String}, \text{String}\}, \emptyset); \tag{4.15}$$

The attributes $A_i$, with $i \in \{2, 3\}$ define the lexeme type of $w$ ($lextype(w)$). The lookup is implemented as such in the first extraction method (manual programming).
Wiktionary based extraction for verbs is defined in equation 4.14. We denote a generic form for the algorithms for verb-based lexemes in equation 4.16. Parameters are applied as defined in table 4.4.

$$\begin{aligned}&\underline{\text{for}}(m_i \in def(w, class(w) := \text{Verb})\ ):\\ &\quad \{\omega\} := \text{words of meaning } m_i;\\ &\quad \exists_{i\in\mathbb{N}_0\ \wedge\ i<|\omega|-\Phi}\ :\ \forall_{j\geq 0\ \wedge\ j<\Phi}\ :\ eq(\omega_{i+j}, \phi_j)\quad \rightarrow\quad lex(w) := \omega_{i+\Phi};\end{aligned} \tag{4.16}$$

$\Phi$ is the number of words of the current pattern and $\phi_j$ is the current word of the current pattern. Each pattern $\varphi_i$ is executed separately.

[13] Example: *Cosmos* (*Sg.*,Ancient Greek). Removal of ‘s’ results in *Cosmo* which is a Scottish male name (variation of the Italian *Cosimo*) and therefore the semantic is highly deferred [http://en.wiktionary.org/wiki/Cosmo].

[14] CSV := Comma-Seperated Values

Table 4.4: Wiktionary based Lexeme Extraction for Verbs

| Pattern $\varphi := \{\phi_1, \phi_1, \dots\}$, $\Phi := \|\varphi\|$ | lextype($w$) |
|---|---|
| [present tense of] | Irregular |
| [simple present indicative form of] | Irregular |
| [simple past tense of] | PastSimple |
| [past participle of] | PastParticiple |
| [present participle of] | PresentParticiple |

Literature: A general definition of lexemes can be found in [CM02].

### 4.2.3 Tempus Extraction

Classification of the tempus (lat., engl. <tense>) attributes a phrase with time information, e.g. the time of happening. We rely on the two mechanisms *Signal Words* (temporal keywords) and *Lexeme Analysis* to get an estimation. To simplify the process (and to reduce uncertainty), tenses $t_i$ are restricted to the set { $t_1$ :=Past, $t_2$ :=Present, $t_3$ :=Future }. For each phrase and tense $t_i$ we define an indicator (:= counter) $\mathcal{I}_i$. We iterate over the words and the appropriate indicator is incremented, if a word of the phrase equals a temporal keyword $kw(t_i)$ (see table 4.5), or its lexeme indicates the tense:

$$\begin{aligned}
&\mathcal{I}_1 := \mathcal{I}_2 := \mathcal{I}_3 := 0;\\
&\underline{\text{for}}(w_i \in p):\\
&\quad \underline{\text{for}}(j \in \{1,2,3\}):\\
&\qquad \mathcal{I}_j := \mathcal{I}_j + \begin{cases} 1, & \text{if } w_i \in kw(t_j) \;\vee\; correlation(\; lextype(w_i),\; t_j\;) \\ 0 & \text{otherwise} \end{cases}\\
&tempus_{estimation}(p) := \max(\mathcal{I}) \;\in\; \{\text{Past}, \text{Present}, \text{Future}\};
\end{aligned} \tag{4.17}$$

The predicate *correlation*(*lextype*, *tense*) is true, if the lexeme type draws inferences of the tempus; e.g. "(lextype) PastSimple ↦ ($t_i$) Past".

Table 4.5: Temporal Keywords by Tense; taken from [Tem]

| Keywords $kw(t_i)$ | $t_i$ |
|---|---|
| { yesterday, last week, last month, this morning, had, have, for, since, lately, already, before, by the time, since } | $t_1$ :=Past |
| { always, usually, often, sometimes, seldom, rarely, never, every day, Mondays, Tuesdays, Wednesdays, Thursdays, Fridays, Saturdays, Sundays } | $t_2$ :=Present |
| { will, going to, will have } | $t_3$ :=Future |

### 4.2.4 Meanings

Wiktionary is used as a tool to extract meanings, since its parsing is a technical side product of parsing the word classes (refer to equations 4.3 and 4.6). While meanings are related to word-classes in Wiktionary and this distinction helps to deal with word class dependent homonyms, they are persisted subordinately to word classes.

## 4.3 Word Frequency

Over the entirety of English-Language texts, a word $w$ has a certain probability to occur. Obviously the inequality

$$\forall_{w \in Corpus} \; : \; \mathit{frequency}(\; \mathit{eq}(w, \text{``a''}) \;) > \mathit{frequency}(\; \mathit{eq}(w, \text{``brobdingnagian''}) \;) \;\; \rightarrow \;\; \text{true}_{(?)}$$

is fulfilled for most texts; with $\mathit{frequency}(w)$, the number of occurrences of the word $w$ in all texts of the Corpus. We define the probability $p(w)$ reciprocally:

$$p(w) := \frac{1}{\mathit{frequency}(w)} \tag{4.18}$$

As later discussed in chapter 6, the word frequency may be used to extract domain-specific words – or more concrete – concepts $\mathcal{C}$ of an ontology $O$. While utilization is discussed in the mentioned chapter, the focus here is on the determination of the frequency on a word basis. For each word $w$ one may distinguish (a) a frequency that $w$ occurs in the examined corpus (in the following "local") and (b) the frequency that $w$ occurs in *all* English texts (in the following "global"). The notation is here:

$$f_{local}(w) := \sum_{w' \in C} \begin{cases} 1 & \text{if } \mathit{eq}(w, w') \\ 0 & \end{cases} \tag{4.19}$$

with $C := \{t_1, t_2, \dots\}$ the input texts, and $\{w'\}$ the list of words from all input texts. Determination of $f_{\text{global}}(w)$ relies on external sources. Theoretically, all exiting textual sources have to be involved; practically a preexisting databases can be used. These databases consist of a cross section of all texts, i.e. a subset of text sources that is representative:

$$f_{global}(w) := \text{Query}(\text{ DB} \in \{\text{BNC}, \;\dots\},\; w \;); \tag{4.20}$$

The implementation mainly uses the BNC (see below) for global frequencies, as well as a small database with 5000 entries for the most frequent words from [`http://corpus.byu.edu/coca/`]. The latter is used to reduce the number of queries to the BNC, and thus decreases run-time.

### 4.3.1 British National Corpus

The British National Corpus (=: BNC) is available at [`http://www.natcorp.ox.ac.uk`] and consists of frequency data, based on a 100 million words corpus of (British) English texts. The consolidation of the substantial number of words and interdisciplinarity implicitly converges the error $e$(BNC, allLanuage)[15] practically to zero, if texts and topics from the 20th century are considered[16]. The accuracy suffices for this project, since we assume that only a few contextfree-words[17] are involved in the process of catchment in the use of language (in the context of natural science texts, i.e. the focus of this project). This implicitly treats all words that are unknown to the BNC, to be domain-specific, as the frequency from the BNC is zero (refer to chapter 6).

Offline access on the BNC is restricted to the United Kingdom[18]. Besides of this location limit, the usage of an online access is freely[19] available at [`http://bncweb.lancs.ac.uk/`]. The URL may be modified[20] to directly receive the needed frequency information. E.g. query of $w :=$ "earth" delivers the result:

[15] The error $e$ is the overall difference between the word-frequencies gained by the BNC, compared to the theoretically word-frequencies from all existing texts.

[16] [Bri14] states that the BNC-corpus "was completed in 1994".

[17] Here: Context-free := Word $w$ is not part of the examined knowledge domain.

[18] University of Oxford Text Archive: `http://www.ota.ox.ac.uk/desc/2554`.

[19] A registration is needed, but free of charge.

[20] Please consider the source code of this project for details.

Table 4.6: BNC Example for $w$ := “earth”

| Query | Result |
|---|---|
| http://bncweb.lancs.ac.uk/cgi-binbncXML/ processQuery.pl?theData=earth&chunk=... | Your query “earth” returned 9194 hits in 1690 different texts (98,313,429 words [4,048 texts]; frequency: 93.52 instances per million words) |

This resulting phrase has to be parsed by a regular language interpreter (Type-3 language $\mathcal{L}_3$). Only a subset of frequency information $\mathcal{F}$ is used at this point; expressed by the tuple:

$$\mathcal{F}' := (\ w,\ p_{global}(w)\ )\ \subset\ \mathcal{F} \tag{4.21}$$

For the word $w$ := earth, an example instance is given as:

$$\mathcal{F}'_{\text{example}} := (\ \text{earth}\ ,\ 9194/98313429\ \approx\ 0.00009352\ ) \tag{4.22}$$

For further information about word frequencies one may consult [AG05].

## 4.4 Extraction of Semantic Relations

Examples for Semantic Relations [Stu] are e.g. Synonyms (=: syn), Antonyms, Hyperonyms (=: hyp), Hyponyms and Meronyms. To get a set of relations for a word $w$ we denote in this work:

$$W := \mathit{hyp}(w) := \text{Hyperonyms of } w,\ \ W := \mathit{syn}(w) := \text{Synonyms of } w, \ldots \tag{4.23}$$

with $W$, a set of words $w_i$. Semantic relations are used in the process of information extraction, as described in chapter 6. Only synonyms and hyperonyms are considered here. While the former may be applied to reduce redundancy, the latter is a basic tool to find hierarchies of concepts ($\mathcal{HC} \in O$).

### 4.4.1 Synonyms

A synonym can formally be defined as:

$$\mathit{eq}(\ \mathit{semantics}(w_1),\ \mathit{semantics}(w_2)\ )\ \ \rightarrow\ \ w_2 \in \mathit{syn}(w_1)\ \Leftrightarrow\ w_1 \in \mathit{syn}(w_2) \tag{4.24}$$

Note that *eq* (:= equals) is sometimes interpreted like “highly correlates”, i.e. the meaning of $w_1$ and $w_2$ matches not exactly. Wiktionary is used as a tool to extract synonyms, since parsing of synonyms is technically similar to parsing meanings (refer to equation 4.3). While synonyms are distinguished by word-classes in Wiktionary, we combine synonyms to a single set here. A further post-processing step must be applied to keep only the significant information. We restrict this to the removal of all parenthesized character sequences as well as omission of special characters. E.g.:

*postprocess*( “(to connect electrically to the earth): (US) ground” ) := “ground”
→ *syn*(earth) := {ground, ... }

### 4.4.2 Hyperonyms

WordNet, a lexical semantic database (developed at Princton University; refer to [Wor14]), groups words in so-called *synsets*, i.e. sets of synonyms and semantic relations. We concentrate on the ability to get hyperonyms[21] *hyp*($w$) for a given word $w$. WordNet can be used offline by a Java-interface. Determination of hyperonyms can be done as follows: (a) Create a synset for $w$ with synset-type “Noun”[22] (b) Iterate over the synset, each retrieve the hyperonyms and (c) Finally

[21] alt. »hypernyms«; often abbreviated to $hr(w)$.

[22] Only nouns are considered here. This restriction will be more clear in chapter 6.

unify the partial hyperonymous solutions. As defined in the first chapter, a hyperonym has the form:

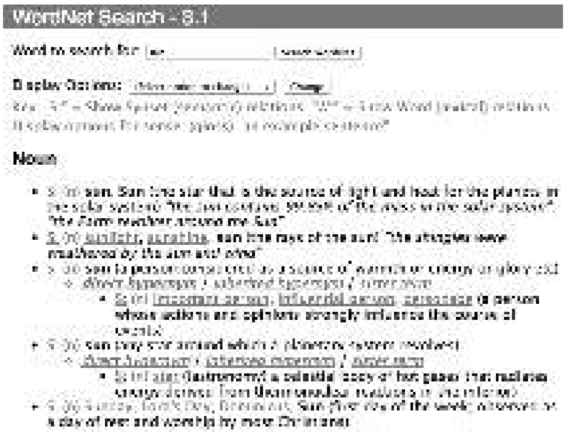
$$isKindOf(w_1, w_2) \;\rightarrow\; w_2 \in hyp(w_1) \;\Rightarrow < w_1, w_2 > \tag{4.25}$$

i.e. we have a hyperonomial relation between the two words $w_1$ and $w_2$, if $w_1$ is kind of a $w_2$. Then $w_2$ is a hyperonym of $w_1$. An example is

$$\text{star} := hyp(\text{sun})$$

We will later on substitute words $w_i$ by concepts $c_i \in \mathcal{C} \subset O$.

Figure 4.4: Example Query for WordNet Entry »sun«



WordNet Search - 3.1

Word to search for: [illegible] [Search WordNet]

Display Options: [illegible] [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

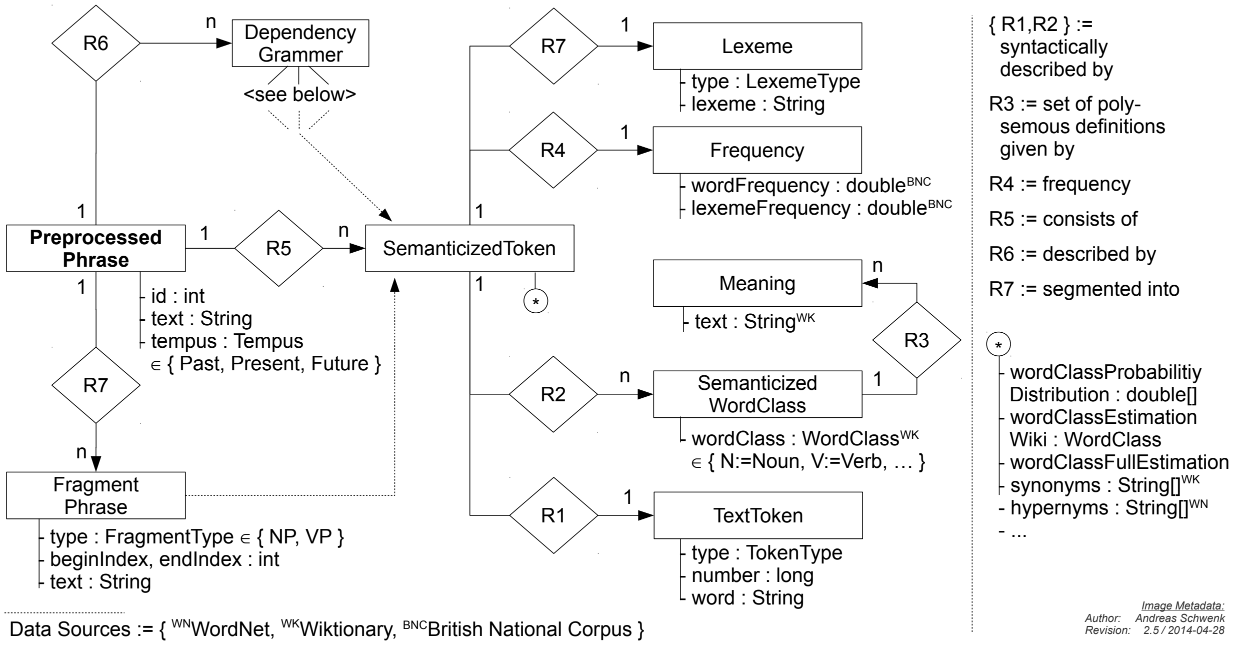
**Noun**

- S: (n) sun, Sun (the star that is the source of light and heat for the planets in the solar system) "the sun contains 99.86% of the mass in the solar system"; "the Earth revolves around the Sun"
- S: (n) sunlight, sunshine, sun (the rays of the sun) "the shingles were weathered by the sun and wind"
- S: (n) sun (a person considered as a source of warmth or energy or glory etc)
  - direct hypernym | inherited hypernym | sister term
    - S: (n) important person, influential person, personage (a person whose actions and opinions strongly influence the course of events)
- S: (n) sun (any star around which a planetary system revolves)
  - direct hypernym | inherited hypernym | sister term
    - S: (n) star ((astronomy) a celestial body of hot gases that radiates energy derived from thermonuclear reactions in the interior)
- S: (n) Sunday, Lord's Day, Dominicus, Sun (first day of the week; observed as a day of rest and worship by most Christians)

## 4.5 Data Modeling

Figure 4.5 depicts the entire ERD for a preprocessed phrase. Note that the Dependency Grammar part is discussed in detail in the next chapter. We treat relations between fragments and semanticized tokens only with indices; therefore there is no relation depicted (see dotted arrow)[23].

Figure 4.5: Entity Relationship Diagram: Phrase


R6
n
Dependency Grammer
<see below>
R7
1
Lexeme
- type : LexemeType
- lexeme : String
R4
1
Frequency
- wordFrequency : double^BNC
- lexemeFrequency : double^BNC
1
Preprocessed Phrase
1
R5
n
SemanticizedToken
1
1
1
*
- id : int
- text : String
- tempus : Tempus
∈ { Past, Present, Future }
Meaning
n
- text : String^WK
R3
R7
R2
n
Semanticized WordClass
1
- wordClass : WordClass^WK
∈ { N:=Noun, V:=Verb, … }
n
Fragment Phrase
R1
1
TextToken
- type : FragmentType ∈ { NP, VP }
- beginIndex, endIndex : int
- text : String
- type : TokenType
- number : long
- word : String
Data Sources := { ^WN WordNet, ^WK Wiktionary, ^BNC British National Corpus }
{ R1,R2 } := syntactically described by
R3 := set of polysemous definitions given by
R4 := frequency
R5 := consists of
R6 := described by
R7 := segmented into
*
- wordClassProbabilitiy Distribution : double[]
- wordClassEstimation Wiki : WordClass
- wordClassFullEstimation
- synonyms : String[]^WK
- hypernyms : String[]^WN
- ...
Image Metadata:
Author: Andreas Schwenk
Revision: 2.5 / 2014-04-28

[23] This avoids the management of further lists in the data structure.

# Chapter 5

# Dependency Grammar Parsing

## 5.1 Introduction

The extraction of the relations $\mathcal{R}_p$ of a phrase $p$ for the domain-specific ontology $O_D$ (with $\mathcal{R} \in O_D$) will be primary based on the Dependency Grammar. In this section we describe a possible process to extract the dependency grammar; input-restricted to the information that are gathered in the process of preprocessing of the last chapter. The main intention is to research the possibility to parse the dependency grammar (only) with methods from Computational Intelligence. Secondary, the amount of external dependencies is kept low, i.e. no specialized dependency databases are used. Thus, the simplicity improves run-time. The accuracy is expected to be less than that of specialized implementations (e.g. compare to *link grammar* e.g. described in [Lin14]). A detailed evaluation can be found in the conclusion of this report.

### 5.1.1 Classical Approaches

Classical approaches make use of large databases, so called *Treebanks*. Treebanks consist of a text corpus that is annotated (mostly manual, i.e. by humans) with syntactic and semantic information. The drawback of Treebanks is (generally) the dependency on licensed work, i.e. fees must be paid for the use. Syntactic Treebanks[1] are subdivided into the underlying type of grammar; mainly phrase structure grammar and dependency grammar[2]. [Wik14b] lists some of the available Treebanks. Filtering[3] restricts the set to { "Prague English Dependency Treebank", "The PARC 700 Dependency Bank", "CHILDES Brown Eve corpus with dependency annotation" }; whereas only the latter is under the terms of *Open Source* and thus available for free of charge. Despite the availability, the intention of this work is to research alternative approaches that waive official Treebanks.

### 5.1.2 A New Neuro-Fuzzy-Approach

As already introduced, Dependency Grammar Parsing relies on the information whether a linking between two given words of a phrase exists: The significant word is called *head*, while the other is called the *dependent*. Treebanks provide relational information for a given input data – or more exact: for a given set of words in their syntactic and semantic relation within a phrase. We later describe that the decision, weather two words are linked together or not, is not depending on the concrete words itself in *this* work. The dependency is contextual, i.e. it relies on the word-class, the position etc. To fulfill the the necessity of *learning* this linking; we make use of the following techniques: In a preceding chapter we introduced both Artificial Neural Networks (=: ANN) and Fuzzy Logic (=: FL). For dependency grammar parsing, we aim at a collaboration of the benefits of both approaches (The purpose of Neuro-Fuzzy in general was first described by J.-S. Roger Jang):

[1] Dependency Grammar Parsing is a syntactic subject matter.

[2] Implicitly, the natural language itself is a further subdivision.

[3] Only consider English language and Dependency Grammar based Treebanks.

- The ANN is used to learn and store uncertain knowledge, based on a below described scheme. Uncertainty can be explained by an arbitrary and variable amount of information provided by the user[4].
- The FL represents a human-readable set of rules. In contrast, the ANN can be seen as a black box. Transforming the neuronal data via a bypass to synthetic Fuzzy rules improves comprehensibility and maintenance of the system.

The combination of both approaches to a hybrid system is, to use the backpropagation learning algorithm and to represent the learned data to a readable form that optionally can be adjusted by hand.

### 5.1.3 Mathematical Representation

To keep the representation-homogeneity of the constructed algorithms, we introduce the following mathematical notation: If word $w_2$ is linked to $w_1$, i.e. $w_1$ is the *head* and and $w_2$ is a *dependent* of $w_1$, we denote:

$$head(w_2) := w_1 \quad \Rightarrow \quad w_2 \in dependent(w_1) \tag{5.1}$$

One word could have more than one dependent. The "quality of link" between two words $w_1$ and $w_2$ is written as:

$$\begin{aligned} eq(head(w_2), w_1) &\rightarrow link(w_1, w_2) \in (0, 1] \\ eq(head(w_2), w_1) &\rightarrow link(w_1, w_2) := 0 \end{aligned} \tag{5.2}$$

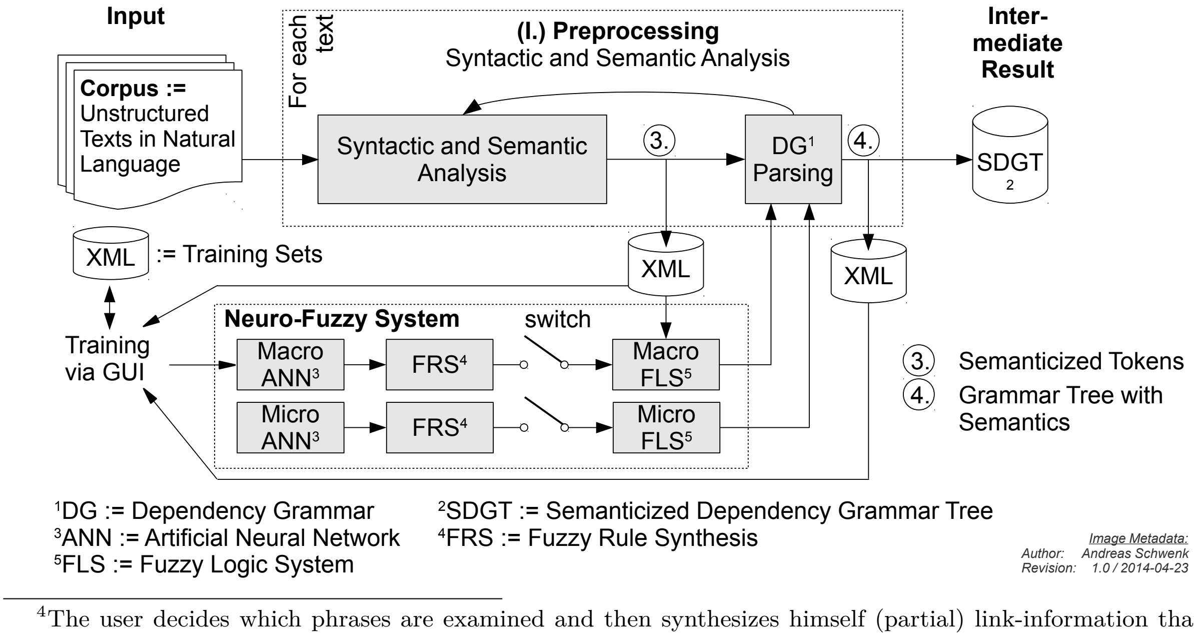
with $eq(,) := equals(,)$. Thus, the value of link is zero, if there is no dependency between the two words; otherwise the amount of link is positive[5].

## 5.2 Design

### 5.2.1 System Integration

Figure 5.1 recapitulates the overview of preprocessing (truncated for the Neuro-Fuzzy related components) with additional details on the Neuro-Fuzzy part:

Figure 5.1: Neuro-Fuzzy System Integration


Input
Corpus := Unstructured Texts in Natural Language
For each text
(I.) Preprocessing
Syntactic and Semantic Analysis
Syntactic and Semantic Analysis
3.
DG¹ Parsing
4.
Inter-mediate Result
SDGT 2
XML := Training Sets
XML
XML
Training via GUI
Neuro-Fuzzy System
switch
Macro ANN³
FRS⁴
Macro FLS⁵
Micro ANN³
FRS⁴
Micro FLS⁵
3. Semanticized Tokens
4. Grammar Tree with Semantics
¹DG := Dependency Grammar
²SDGT := Semanticized Dependency Grammar Tree
³ANN := Artificial Neural Network
⁴FRS := Fuzzy Rule Synthesis
⁵FLS := Fuzzy Logic System
Image Metadata:
Author: Andreas Schwenk
Revision: 1.0 / 2014-04-23

[4] The user decides which phrases are examined and then synthesizes himself (partial) link-information that will result in training-sets.

[5] In contrast to this definition, the GUI will later on use a different scaling $\in [0, 1]$. The interpretation will be: $[0, 0.5) \rightarrow$»bad link«; $[0.5, 1] \rightarrow$»good link«.

Accordingly, the intermediate XML-databases – in form of tokens with semantics for each phrase $p$ – are laid to the Graphical User Interface (=: GUI) as requested: The user retrieves an input and a feedback that is specified in the next subsection. The Neuro-Fuzzy-sequence is subdivided into:

1. Particular[6] Training-sets are manually generated and then stored into files in the XML format. Training the Artificial Neural Network is done via the Backpropagation Learning Algorithm and may include all previous training sets from previous runs. We distinguish a training on a »macro« and a »micro« basis (see below).
2. The Fuzzy Rule Synthesis (=: FRS) component transforms the uncertain data into human-readable and thus NLP-based fuzzy rules.
3. Updating is finally done via "closing the switch" (refer to figure 5.1) and implies substitution of (possibly existing) previous fuzzy rules.
4. Determination of link$(w_1, w_2)$ is finally done by the Fuzzy Logic System (=: FLS).

### 5.2.2 Training

Figure 5.2 illustrates the Graphical User Interface (=: GUI) for training. A single phrase, including the word classes and fragment types, is each depicted. Since the goal is a *hierarchical* parsing of the dependency grammar; each a training on the word base and the fragment base is possible. Hierarchical parsing has the benefit to implicitly simplify the process by considering different types of granularity:

1. Micro Dependency Grammar Learning (=: Micro-DGL):
   Determines the link for each two words $w_1$ and $w_2$:

   $$\mathit{link}(w_1, w_2) \tag{5.3}$$

   For the example phrase, one may link the two words ("the earth") to 1.0 (» good« link), i.e.

   $$\mathit{link}(\underbrace{\text{earth}}_{\text{Noun}}, \underbrace{\text{the}}_{\text{Det}}) := 1.0 \quad \Leftrightarrow \quad \boxed{\text{the}} \leftarrow \boxed{\text{earth}} \tag{5.4}$$

   This is valid, since "earth" is the *head* and "the" a (here: the only) *dependent*.

2. Macro Dependency Grammar Learning (=: Macro-DGL):
   The Macro-DGL is based on phrase fragments $f_i$:

   $$\mathit{link}(f_1, f_2) \tag{5.5}$$

   For the example phrase, one may e.g. link:

   $$\mathit{link}(\ \text{VP(is)},\ \text{NP(the earth)}\ ) := 1.0 \quad \Leftrightarrow \quad \boxed{\text{the earth}} \leftarrow \boxed{\text{is}} \tag{5.6}$$

   This is valid, since "is" is the *head* and "the earth" the dependent[7].

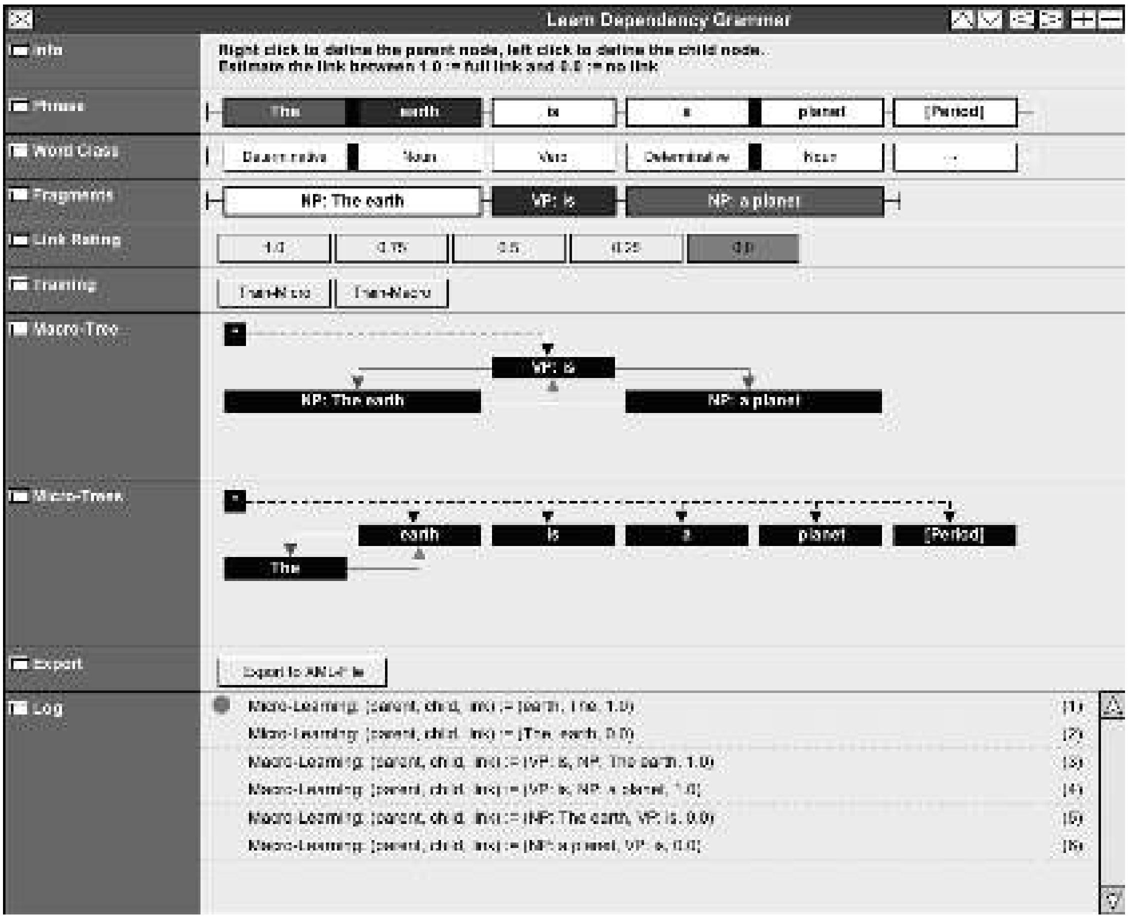
[6] The user may choose which phrases should be used for training. It is also possible to only train subphrases.

[7] The set of examples above can be treated as *positives*, since the link is always set to 1.0. Effectiveness in determining the link within the later described parsing process can only be learned, if also lower link-values of less that one (*link* $< 1$) are provided. We define $\{1.0, 0.5, 0\} \mapsto$ {»good«,»neutral«,»bad«}. A »bad« case would e.g. to set a verb to be the *dependent* and a "linked" noun to be the *head*.

### 5.2.3 Graphical User Interface

Figure 5.2 shows the training in progress:

Figure 5.2: Graphical User Interface for Dependency Grammar Learning



Right click to define the parent node, left click to define the child node.
Estimate the link between 1.0 := full link and 0.0 := no link

```
Micro-Learning: (parent, child, link) := (earth, The, 1.0)            (1)
Micro-Learning: (parent, child, link) := (The, earth, 0.0)            (2)
Macro-Learning: (parent, child, link) := (VP: is, NP: The earth, 1.0) (3)
Macro-Learning: (parent, child, link) := (VP: is, NP: a planet, 1.0)  (4)
Macro-Learning: (parent, child, link) := (NP: The earth, VP: is, 0.0) (5)
Macro-Learning: (parent, child, link) := (NP: a planet, VP: is, 0.0)  (6)
```

Table 5.1: Dependency Grammar Learning – GUI Options

| Panel | Description |
|---|---|
| Phrase | List of words. The selection is described in the info panel. |
| Word Class | Word class of each word. |
| Fragment | Nominal and Verbal Phrases. |
| Link Rating | $\{1.0, \ldots, 0.5, \ldots, 0\} \mapsto \{$ »good«, ..., »neutral«, ..., »bad« $\}$. |
| Training | Creates a training set; based on the current selection. |
| {Micro,Macro}-Tree | Feedback of the learning process. Top-Down arrows indicate a »good« link. Bottom-Up arrows indicate a »bad« link. |
| Export | Writes training data to XML files (refer to listings 5.1 and 5.2). |
| Log | Logging information. |

Listings 5.1 and 5.2 show the export-format for each a micro and macro dependency training example.

Listing 5.1: Micro-DG Training Set Example

```
<TS index="3">
  <Type t="Micro"/>
  <Phrase text="The␣earth␣is␣a␣planet."/>
  <WordClasses wc="Determinative␣Noun␣Verb␣Preposition␣Noun␣Period"/>
  <Data parentIndex="1" parentName="earth" childIndex="0" childName="the" link
      ="1.0"/>
  <TimeStamp ts="Thu␣Jan␣30␣00:32:57␣CET␣2014"/>
</TS>
```

Listing 5.2: Macro-DG Training Set Example

```
<TS index="4">
  <Type t="Macro"/>
  <Phrase text="[The␣earth]␣[is]␣[a␣planet][.]"/>
  <WordClasses wc="Noun␣Verb␣Noun␣Period"/>
  <Data parentIndex="1" parentName="[is]" childIndex="0" childName="[The␣earth
      ]" link="1.0"/>
  <TimeStamp ts="Thu␣Jan␣30␣00:32:48␣CET␣2014"/>
</TS>
```

## 5.3 Algorithm

### 5.3.1 Artificial Neural Networks for Supervised Learning

Within the training-set generation via the Graphical User Interface, we can gather structured data for link information on a word and/or fragment basis. The next step is to transform the data range, to provide the information to the neurons of the input layer of the ANN:

$$\text{LinkData} \mapsto \text{InputData(ANN)} \subseteq R^{M_0}, \quad R = [0,1] \in \mathbb{R} \tag{5.7}$$

with $M_0$ the number of input neurons and $R$ ($\neq \mathcal{R}$), the input vector with $M_0$ elements that are each restricted to $[0,1] \in \mathbb{R}$ (note[8]). As a design decision, we chose *not* to learn explicit words (respectively phrase fragments), but rather we learn the *environmental* data for each atom[9] =: its context. The context is described by:

- Atom-type := class of the *dependent*:
  Map the word-class resp. the type of the fragment phrase[10] $\in \{\text{NP}, \text{VP}\}$ to $[0,1] \in \mathbb{R}$:

  $$\begin{aligned} x_1' &:= \mathit{ordinal}(\mathit{class}(w_i) \in \{\text{Noun}, \text{Verb}, \dots\}) - 1 \quad \in \quad \{0,1,\dots\} \in \mathbb{N}_0 \\ x_1 &:= x_1' \ / \ \max\{X_1'\} \quad \in \quad [0,1] \in \mathbb{R} \end{aligned} \tag{5.8}$$

  $\max\{X_1'\}$ is here set to 10. For the fragment consideration, "class($w_i$)" must be changed accordingly: All word classes in the phrase that could not be tagged to {VP,NP} are kept from the micro system.

- Relative position between *head* and *dependent*:
  The position for each word $w_i$ in a phrase can be measured from left to right:

  $$\mathit{pos}(w_i) \ \in \ [0, |p|-1] \in \mathbb{N}_0 \tag{5.9}$$

  The relative position between the *head* and the *dependent* is thus:

  $$\mathit{pos}'_{\text{rel.}} := \mathit{pos}(w_{\mathit{dependent}}) - \text{pos}(w_{\mathit{head}}) \ \in \ [-|p|+1, |p|-1] \in \mathbb{N}_0 \tag{5.10}$$

[8] Refer to the introduction to ANNs. The used internal functions – especially the activation function *sigmoid* – only have range $[0,1] \in \mathbb{R}$.

[9] Atom for Macro-Parsing =: Word; Atom for Micro-Parsing =: Phrase-Fragment.

[10] We internally represent {NP,VP} as {Noun,Verb}.

$pos'_{rel.}$ must be normalized to keep the bounds in the range $[0, 1]$:

$$x_2 := pos_{rel.} := \underbrace{\text{median}\left(-1,\ \frac{pos'_{rel.}}{scalingfactor},\ 1\right)}_{\in[-1,1]\in\mathbb{R}} \cdot 0.5 + 1 \tag{5.11}$$

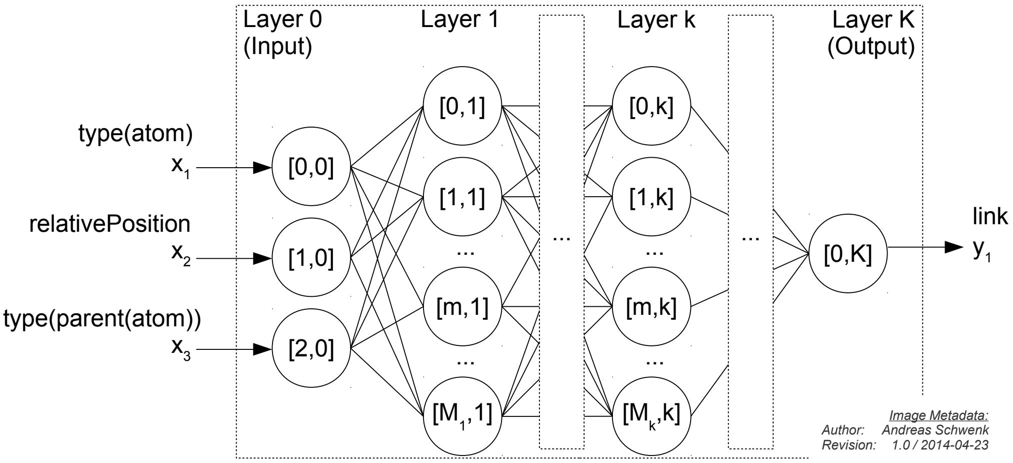
The scaling factor is empirically chosen to be 10; cases with $|pos_{rel.}| > 10$ are treated as 10.

- Type of the atom's parent := type of the *head*:
  Refer to the calculation of "Atom-type" and replace $w_i$ with $w_j$[11]. The destination variable is $x_3$.

We finally get a tuple $TS := (X, Y) := (\{x_1, x_2, x_3\}, y_1)$ for the training set.

#### 5.3.1.1 Determining the Dimensionality of the ANN

Figure 5.3: Structure of the ANN



The set $X := \{x_1, x_2, x_3\}$ represents the values for the input-neurons of the ANN and $Y := \{y_1 := link\}$ is the output. Refer to figure 5.3 to get a schematic overview. Some generic variables are not yet covered:

- $K + 1$ := the number of layers (Layers are indexed with $[0, K] \in \mathbb{N}_0$).
- $\{M_1, M_2, \ldots, M_{K-1}\}$ := the number of neurons per hidden layer.

Since Backpropagation Learning is applied, the number of needed neurons cannot be derived in a trivial manner (Compare e.g. to [LGT96]). Instead of a mathematical estimation, we postulate the following iterative algorithm to increase the dimensionality up to a level where the remaining error (:= the difference between the real link value and the link value that the ANN calculates) is below a given threshold. Note that this procedure is of empirical nature. We define $H := K - 1$, the number of hidden layers, that is all layers that are neither an input layer nor an output layer:

```
(1.) $H := 1; \quad \forall_i \ : \ M_i := 3;$
(2.) do :
        $error$ := Perform Backpropagation learning with Training Set Data $TS$;
        $\forall_i : M_i := M_i + 1;$
        if $(M_1 > T_1)$ then :
            $\forall_i : M_i := 3; \quad H := H + 1 \quad (\rightarrow |M| := |M| + 1);$
        end if;
     while $(error < T_2)$;
                                                                    (5.12)
```

[11] eq(head($w_i$),$w_j$) must be fulfilled.

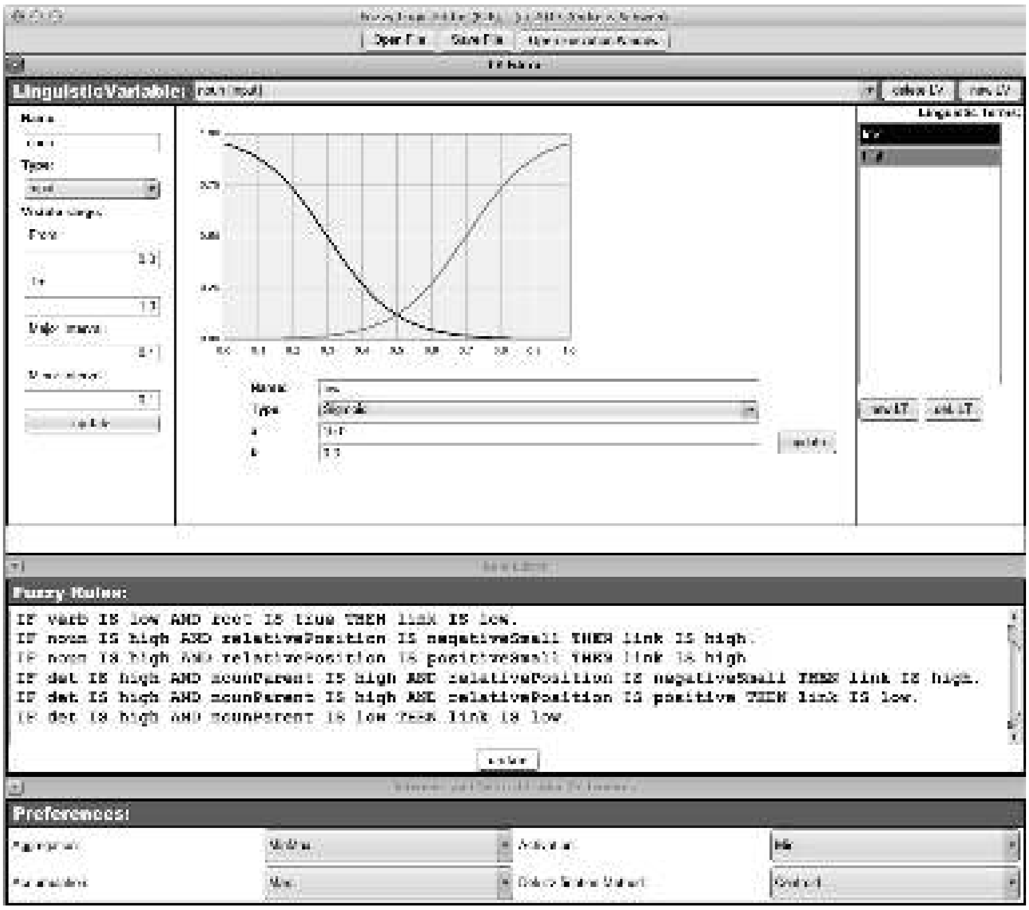
The two thresholds may e.g. be set to $\{T_1, T_2\} := \{8, 0.01\}$. In this case, we start with one hidden layer that consists of initially three neurons. As long as the total error is greater or equal to 0.01, the number of neurons per hidden layer is incremented. The choice of the value 0.01 is sufficient, since the link must not be very precise and in addition, we get a relaxation for uncertain training sets. The latter may occur, if the user produces contradictory training data. The number of hidden layers is incremented, when 8 neurons per hidden layers are not sufficient; additionally a reset of neurons per hidden layer is done, i.e. set to 3. Remark[12].

Furthermore should the network size be bounded. There may exist data that does not causes a learning-convergence within reasonable time. The training set-XML-files should be reviewed and adjusted as a work-around.

### 5.3.2 Fuzzy Logic to Clarify the Parsing Behavior with Natural Language Rules

To ease the handling of Fuzzy Logic, a Fuzzy Logic Editor (=: FLE) and the underlying Fuzzy Logic System (=: FLS) were implemented in Java. Figure 5.4 shows an example Screenshot.

Figure 5.4: Fuzzy Logic Editor – Screenshot



```
IF verb IS low AND root IS true THEN link IS low.
IF noun IS high AND relativePosition IS negativeSmall THEN link IS high.
IF noun IS high AND relativePosition IS positiveSmall THEN link IS high
IF det IS high AND nounParent IS high AND relativePosition IS negativeSmall THEN link IS high.
IF det IS high AND nounParent IS high AND relativePosition IS positive THEN link IS low.
IF det IS high AND nounParent IS low THEN link IS low
```

The usage is more or less self-explaining: One may create a set of Linguistic Variables (=: LVs) and relating Linguistic Terms (=: LTs). Preferences remain as set (refer to [Bar13] for meanings) and the synthesis of rules is described below.

[12] Starting with a high dimensionality is not an option, since the amount of calculations does not increase linearly. Weights $w$ are adjusted as follows:

$$\forall_{\text{epochs } e} : \forall_{\text{trainingSets } t_s} : \forall_{\text{layers } l} : \forall_{\text{neuronsInLayer } n} : \forall_{\text{inputs } i} : \text{adjust } w_{n,l,i} \tag{5.13}$$

#### 5.3.2.1 Define the Linguistic Variables

Definition of the Linguistic Variables (=: LVs) requires to represent all the information from the ANN. Table 5.2 summarizes the LVs briefly[13].

Table 5.2: Linguistic Variables LV ∈ FLS

| LV | type(LV) | LTs | type(LT) |
|---|---|---|---|
| noun* | input | {low,high} | {Sigmoid(10,0.3), Sigmoid(-10,0.7)} |
| nounParent* | input | {low,high} | {Sigmoid(10,0.3), Sigmoid(-10,0.7)} |
| position | input | {positiveSmall, positiveLarge} | {Trapezoidal(-1,-1,5,15), Trapezoidal(0,15,40,100)} |
| relPosition | input | {NS,PS,Z,PL,NL,P} | {Trapezoidal(. . . )} |
| link | output | {low,medium,high} | {Singleton(0.0),Sglt.(0.5),Sglt.(1.0)} |

#### 5.3.2.2 Integration of the FLE

The Fuzzy Logic Editor is associated with the ontology extraction system by an XML-structured file. This way, the editor can be kept as an independent component. Listing B.1 in the appendix shows an excerpt of an export of the preferences. The final task of the FLS is given as:

$$\begin{aligned}
&\boxed{link(w_1, w_2)} && := \text{FLS}(\vec{v}_{\text{in}}) \\
&\vec{v}_{in} && := \vec{v}_{in,1} \cup \vec{v}_{in,2} \cup \vec{v}_{in,3} \\
&\vec{v}_{in,1} && := \{\ p(class(w_1), \text{Noun}),\ p(class(w_1), \text{Verb}),\ \dots\ \} \\
&\vec{v}_{in,2} && := \{\ p(class(w_2), \text{Noun}),\ p(class(w_2), \text{Verb}),\ \dots\ \} \\
&\vec{v}_{in,3} && := \{relativePosition(w_1, w_2)\}
\end{aligned} \tag{5.14}$$

$p(class(w_i), \text{<WordClass>})$ is the probability $\in [0, 1] \in \mathbb{R}$ that the word-class of word $w_i$ is equal to <WordClass>. Please refer to the chapter about preprocessing, form further information about the probability of word-classes.

**Example:** If the link(earth,the) for the phrase $p\ := \vdash$ The $\rightarrow$ earth $\rightarrow$ *is* $\rightarrow$ a $\rightarrow$ planet $\dashv$ should be determined, the following (partial defined) input vector $\vec{v}_{in}$ for the FLS would ideally be constructed[14]:

Table 5.3: Example Input-Vector for the FLS

| LV-input | Example-value |
|---|---|
| $p(class(the), \text{Det})$ | 1 |
| $p(class(earth), \text{Noun})$ | 0.8 |
| $p(class(earth), \text{Verb})$ | 0.2 |
| $relativePosition$ | 1 |

Assuming that fuzzy rules would already exist, the FLS would calculate the output *link* to a value of e.g. 0.85; i.e. the dependency $\boxed{\text{The}} \leftarrow \boxed{\text{earth}}$ is treated to be true for 85 %.

[13] Legend: relPosition := relative position, NS := negative small, PS := positive small, PL := positive large, NL := negative large, P := positive. $Sigmoid(x, a, b)$ is defined as $\frac{1.0}{1.0+e^{a\cdot(x-b))}}$.
$Trapezoidal(x, a, b, c, d)$ is defined as $x \geq a ? \frac{x-a}{b-a} : (x \geq b \wedge x \leq c ? 1 : (x \geq c \wedge x \leq d ? 1 - \frac{x-c}{d-c} : 0))$.
$Singleton(x, a)$ is defined as $x = a ? 1 : 0$.
Example curves for Sigmoid can be observed in figure 5.4. Each LV that is in table 5.2 attributed with "*", can be copied for all other word-classes with unaltered definitions.

[14] All non-given values are assumed to be zero. The word "earth" is a homonym and has different word-classes. Semantics can be defined as: (a) "Earth" as a noun, e.g. earth := ground. (b) "Earth" as a verb, e.g. to earth an electrical circuit.

### 5.3.3 Synthesis of the Fuzzy Rules from the Neuronal Data

The last subsections described, how the ANN is used for knowledge representation of uncertain knowledge for dependency training-sets; and how the FLS can be used to calculate the link, given an input vector. Now we describe the core of the FLS: A set of fuzzy-rules is used to describe the internal calculation with natural language rules. The regular structure of such a rule $R_i$ was introduced in the basic-chapter:

$$R_i \ := \text{IF} <\text{premise } p> \text{THEN} <\text{conclusion } c>. \tag{5.15}$$

Listing 5.3 shows some examples for manually formulated rules:

Listing 5.3: Example Fuzzy Rules

```
IF det IS high AND nounParent IS high AND relativePosition IS negativeSmall THEN link IS high.
IF det IS high AND nounParent IS high AND relativePosition IS positive THEN link IS low.
IF det IS high AND nounParent IS low THEN link IS low.
```

The first line can be translated into "If the word (here: the *dependent*) is to a high probability of the word-class 'determiner', and the *head* is likely to be a 'noun' with a high probability, and the relative position between head and dependent is 'negative-small' (:= not much below zero); then link(head,dependent) is high". Compare this formulation to the last example ("The earth").

Despite a manually formulation of all rules is possible, this would be a time-consuming task for all combinations of word-classes and relative positions. We now use the trained ANN to synthesize these rules and write them into a XML-file. A further fine-tuning in form of a manual post-processing step would be possible; since the human-readable set of fuzzy rules is highly in contrast to the black-box-like ANN. The trained ANN can be stimulated with a sequence of combinations of $X := \{x_1, x_2, x_3\}$ to force a calculation of the link $y_1$:

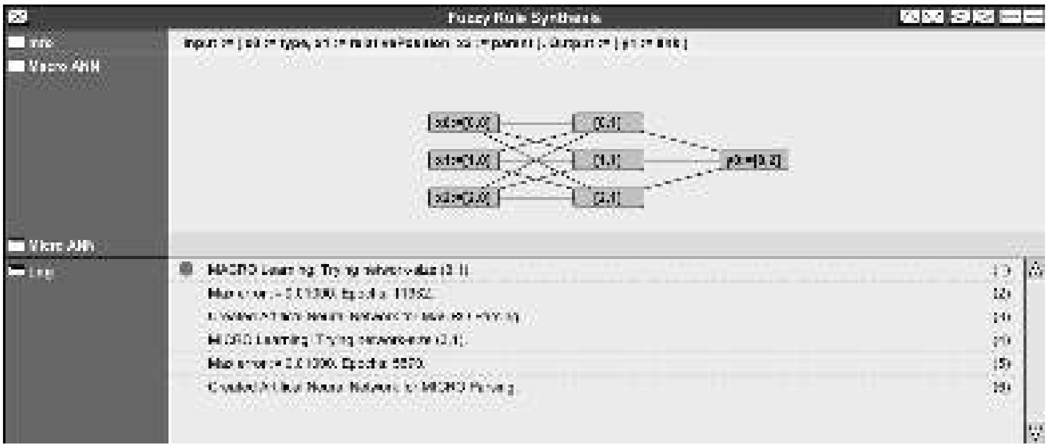
```
for ($wc_p \in \{\text{Noun}, \text{Verb}, \text{Adjective}, \dots\}$) :
  for ($wc_i \in \{\text{Noun}, \text{Verb}, \text{Adjective}, \dots\}$) :
    for ($rel_{pos} \in \{-5, -2, 2, 5\}$) :
      $y_1 := \text{ANN}(\ x_1 := convert_1(wc_p),\ x_2 := convert_1(wc_i),\ x_3 := convert_1(rel_{pos})\ );$
      Rule $r :=$ IF $wc_p$ IS high AND $wc_i$ IS low AND rel.Pos IS $rel_{pos}$
                THEN link IS $convert_2(y_1)$;
    end for;
  end for;
end for;
```

(5.16)

with $wc_p$ the current parent word-class (:= *head*) and $wc_i$ the current child word-class (:= *dependent*). The conversion functions $\{convert_1, convert_2\}$ transform the values either to the ANN or FLS representation. This generation procedure has to be done for each the Micro and Macro part.

Figure 5.5 shows an example training process. The depicted Macro-ANN is quiet small, due to a very low amount of training data:

Figure 5.5: Graphical User Interface for Fuzzy Rule Synthesis



Input := { x0 := type, x1 := relative position, x2 := parent }, Output := { y0 := link }

(1) MACRO Learning: Trying reference value (2.1)
(2) Max error = 0.01000, Epoch = 11362.
(3) Created [illegible] Neural Networks for [illegible]
(4) MICRO Learning: Trying reference value (2.1).
(5) Max error = 0.01000, Epoch = 5570.
(6) Created [illegible] Neural Networks for MICRO Parsing.

### 5.3.4 Build the Grammar Tree

Given the ability to calculate $\mathit{link}(w_1, w_2)$ for every two words $w_1$ and $w_2$ (resp. to calculate the link for any two fragments $f_1$ and $f_2$), we have to decide how the parsing process is structured. [Cov01] describes several approaches. Some ideas have been taken from this source to derive the following – adjusted – algorithm:

The idea is to generate a graph $G = (V, E)$ that is a tree $T = G$, i.e. no cycles in $G$ exist. We distinguish two steps:

1. Find the root node $v_{root} \in V$. The root node is a word $w$ that fulfills $eq(class(w), \text{Verb})$, since the main verb is significant in Dependency Grammar.
2. Recursively extend the previous tree $T$: mount all other words $w_d$ to each a parent node with the constraint: $\forall_{w_d \in p} : maxValue := \max\{link(w_h, w_d), maxValue\}$, with $w_d$ an unprocessed node (word of phrase $p$; a *dependent* =: d) as a link candidate and $w_h$ a potential parent node (*head* =: h). An edge $e := (w_h, w_d) \in E$ is added to $T$, in case $eq(link(w_h, w_d), maxValue)$.[15]

More precisely, we denote the complete algorithm as follows:

```
(1.)   $G = (V, E) \;:=\; (\; \{w_1, \ldots, w_{|p|}\},\; \emptyset \;);$
(2.)   $\varepsilon \;:=\; GF(2)^{|p|} \;:=\; [0\ 0\ \ldots\ 0], \quad |\varepsilon| \;:=\; |p|;$
(3.)   $\exists_{r \in [1,|p|]} \;:\; eq(\; link(\emptyset, w_r),\; \max\{\; link(\emptyset, w_1),\; link(\emptyset, w_2), \ldots,\; link(\emptyset, w_{|p|})\}\;)$
       $\wedge\; eq(class(w_r), \text{Verb});\;\; \varepsilon_r := 1;$
(4.)   $L_{|p| \times |p|}$ with $l_{i,j} \in L \in \mathbb{R} \;\; (\{i, j\} \in \mathbb{N} \;\wedge\; \{i, j\} \leq |p|);$
(5.)   while ( $\exists_{i \in [1,|\varepsilon|]} \;:\; eq(\varepsilon_i, 0)$ ) :
(5.1)    for ( $d \in [1, |p|]$ ) :
           for ( $h \in [1, |p|] \;\wedge\; eq(\varepsilon_h, 0)$ ) :
             $l_{h,d} := \begin{cases} link(w_h, w_d) & \text{if } \left\{ \begin{array}{l} (d < h \;\vee\; d > maxCP(w_h \in V)) \;\wedge \\ (d > h \;\vee\; d < minCP(w_h \in V)) \end{array} \right\} \\ -1 & \text{otherwise;} \end{cases}$   (5.17)
           end for;
         end for;
(5.2)    $\mathcal{M} := \max\{l_{h,d}\}, \quad$ with $\; h \in [1, |p|] \wedge eq(\varepsilon_h, 1), \quad d \in [1, |p|] \wedge eq(\varepsilon_d, 0);$
(5.3)    $e \;:=\; (\; h(\mathcal{M}),\; d(\mathcal{M})\;);$
         $E \;:=\; E \;\cup\; \{e\};$
         $\varepsilon_h \;:=\; \varepsilon_d \;:=\; 1;$
       end while;
(6.)   $|V| := |E| + 1, \quad$ i.e. $G$ is a tree;
```

[15] $eq(x, y)$ is the predicate $equals(x, y) := x = y$.

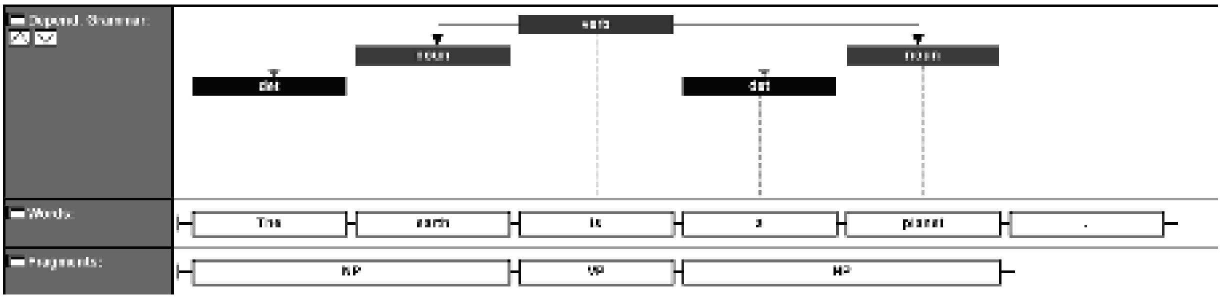
For simplicity reasons, all indices of $p$ are $\in [1, |p|] \in \mathbb{N}$. The implementation requires to transform the range to $[0, |p|-1] \in \mathbb{N}_0$. Furthermore $h :=$ head index and $d :=$ dependent index. $maxC_{idx}(v_i) :=$ maximum index of all child-nodes of node $v_i$: Step (5.1) enforces the graph to be "visually planar". $\varepsilon :=$ List of already processed words $w$ of a phrase $p$ (A Galois field with elements $\in \{0, 1\}$). $|p| :=$ number of words in the phrase. $L :=$ link-matrix. Predicate $eq(x, y) := equals(x, y) := x = y$.

**Hierarchical Parsing:** Hierarchical parsing enhances the algorithm:

- (Def. *Macro parsing*:) For a parsing based on fragment phrases $f_i \in \{\text{NP}, \text{VP}, \dots\}$ we replace each $w_i$ by $f_i$. The length of a phrase, i.e. the number of words, is substituted to the number of fragments: $|p| \mapsto |f|$.
- (Def. *Hybrid parsing*:) The complete calculation is first performed on a fragment base and then – for every $f_i$ – the parsing on a word basis is done on the subphrase for $f_i$; ranged by $[w_j, \dots w_{j+|f_i|}]$.

### 5.3.5 Example

Figure 5.6: Dependency Grammar Parsing Example

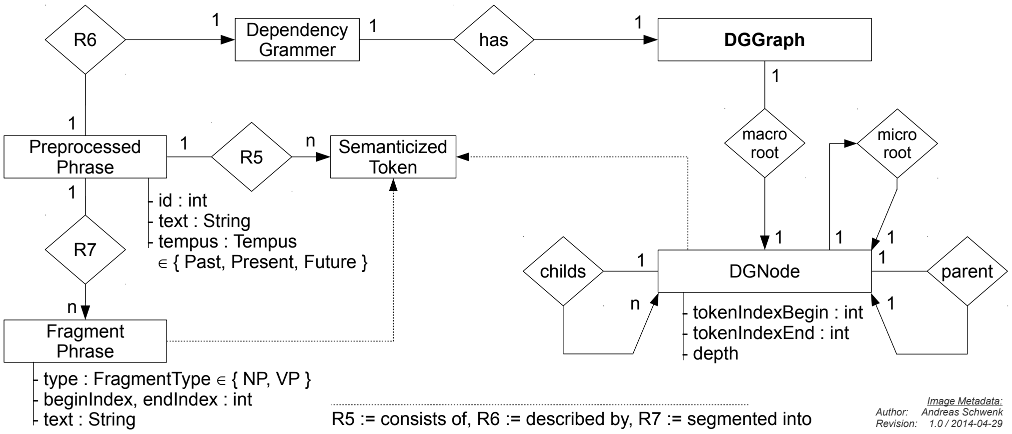


## 5.4 Data Modeling

Figure 5.7 extends figure 4.5 to the Dependency Grammar that is implemented in a tree structure. One of the nodes is declared as the root for the macro-graph[16]. Within the tree-structure of a macro-graph, a micro-graph can be instantiated. The constraint, that a node of the micro-graph cannot recursively instantiate another micro-graph, holds.

The reference from a node to an information element is context-sensitive, so we build associations only with indices to words $w_i$ (here: semanticized tokens) resp. fragements $f_i$; the connection is constructed at run-time.

Figure 5.7: Partially Entity Relationship Diagram: Phrase with Dependency Grammar



[16] Note that for this node parent($n_{\text{root}}$) is equal to $\emptyset$.

# Chapter 6

# Information Extraction

The ultimate goal in the process of information extraction is the synthesis of the subset ontology $O'$ for a knowledge domain $D$:

$$O'_D = (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \emptyset) \subset O_D = (\mathcal{C}, \mathcal{R}, \mathcal{HC}, \mathcal{A}0), \quad \Rightarrow \ \mathcal{A}0 := \emptyset \tag{6.1}$$

## 6.1 Word Frequency Analysis (Concepts $\mathcal{C}$)

### 6.1.1 Moving Toward a First Estimation of Concepts

As described in the chapter about preprocessing, we can construct the following quintuple for every word $w \in$ Corpus (in this context we define $w := lex(w)$, since candidates for concepts are examined); containing relevant frequency information $\mathcal{F}$:

$$\mathcal{F} := (w\ , class(w),\ f_{local}(w),\ f_{global}(w),\ \tilde{w}) \tag{6.2}$$

We denote $class(w)$ as the word-class of the word $w$, $f_{local}$ the frequency (:= number of occurrences) of the word $w$ in all examined texts $T_{local}$, $f_{global}$ the frequency of the word $w$ in the entirety of English-language texts[1] $T_{global}$, and $\tilde{w}$ the so-called *weirdness* – a correlation measure between the local and global frequency. The idea of the latter is taken from [AG05], and is defined as a ratio:

$$\tilde{w}' := \frac{f_{local}(w)}{f_{global}(w)} \tag{6.3}$$

Since the BNC neither contains all requested words nor their lexemes, we introduce a practical adjustment to finally calculate the weirdness $\tilde{w}$; given $\tilde{w}_{\max} \geq \max\{\tilde{W}\} + 1$. E.g. $\tilde{w}_{\max} := 1000000$:

$$\tilde{w} := \begin{cases} \tilde{w}' & \text{if } \ f_{global} > 0 \\ \tilde{w}_{\max} & \text{otherwise} \end{cases} \tag{6.4}$$

To get a first estimation for the concepts (=: $\mathcal{C}_{\text{Estimation}}$), we choose all words that have a word-class that is presumably[2] equal to *noun*, as well have a significant weirdness:

$$\mathcal{C}_{Estimation}(\supseteq \mathcal{C}) := \{(c := w) \in \mathcal{F} \ \mid \text{gT}(c, \Theta) \ \wedge \ eq(c, \text{Noun}))\} \tag{6.5}$$

The abbreviations for the two predicates are defined as gT := greater than, and eq := equals. The threshold $\Theta$ must be set empirically and may depend on the examined knowledge domain $D$.

[1] Based on the subset of the British National Corpus =: BNC.

[2] Remember the remaining uncertainty in word class determination.

### 6.1.2 Reduction of Redundancy

Under the involvement of all synonyms $syn(c \in \mathcal{C})$ for each $c$, we reduce (in best cases *remove*) the redundancy and improve the first estimation:

$$\mathcal{C}_{Estimation} \mapsto \mathcal{C} \tag{6.6}$$

#### 6.1.2.1 First Approach

Remove synonymous concepts:

$$\begin{array}{ll} (1.) & \mathcal{C} := \emptyset \\ (2.) & \underline{\text{for}}\ (c_1 \in \mathcal{C}_{est.}) : \\ & \quad \mathcal{C} := \mathcal{C} \cup c_1; \quad \mathcal{C}_{est.} := \mathcal{C}_{est.} - \{syn(c_1)\}; \\ & \underline{\text{end for}}; \end{array} \tag{6.7}$$

This has the disadvantage that we lose information for further processing steps.

#### 6.1.2.2 Second Approach

In case that a synonym is detected, we add a relation $r \in \mathcal{R}$:

$$\mathcal{R} := \mathcal{R}\ \cup\ \{\ r := \text{synonym}(c_1, c_2)\ \} \tag{6.8}$$

The implementation uses this approach.

### 6.1.3 Example and First Evaluation

Table 6.1 shows four words $w \in T$, i.e. four words of the corpus, and its resulting participance in the set of concepts $\mathcal{C}$. Note that the local frequency does not differ for the first three words because each appearance is one of 2010 evaluated words. The word *participle* would also be $\in \mathcal{C}$ in case the threshold is greater than 2000; but otherwise it is not a member of the chosen knowledge domain $D$ := "The Universe". This side effect is not handled in the implementation and thus $\mathcal{C}$ must be furthermore seen an estimation and varies from the real set of concepts. If the amount of parsed texts is very large, the uncertainty can be reduced[3].

Table 6.1: Example Frequency Information $\mathcal{F}$

| $w$ | class($w$) | $f_{local}(w)$ | $f_{global}(w)$ | $\tilde{w}$ | $w \in \mathcal{C}$? |
|---|---|---|---|---|---|
| cosmogony | noun | 0.000497512 | 0 | 1000000 | true |
| geocentric | adjective | 0.000497512 | 0.0000001 | 4975.124 | false |
| participle | noun | 0.000497512 | 0.0000002 | 2487.562 | (true) |
| if | conjunction | 0.001492537 | 0.0020745 | 0.719 | false |

[3] All non-domain specific nouns $w$ occur randomly.

## 6.2 Dependency Grammar Analysis (Relations $\mathcal{R}$)

The use of parsing the Hierarchical Dependency Grammar of each phrase is to create the relations $\mathcal{R} \in O = \{r_1, \ldots, r_{|\mathcal{R}|}\}$. We assume that is sufficient[4] to only handle binary relations given the form:

$$r \in \mathcal{R} \wedge eq(\mathit{fragmenttype}(r), \mathrm{VP}) := r_{name}(c_1 \in \mathcal{C}, c_2 \in \mathcal{C}) \tag{6.9}$$

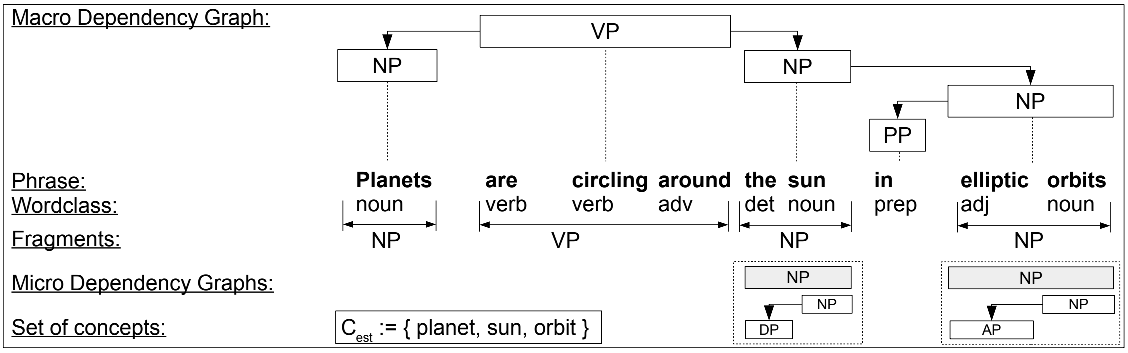
and in addition enforce:

$$c_i \in \mathcal{C} \wedge eq(\mathit{class}(c_i), \mathrm{Noun}) \tag{6.10}$$

i.e. $r$ is a verbal phrase (=: VP) that itself is not atomic; but this (advantageously) compensates the lack of the contradistinction of $c$ – with $\{c \in \mathcal{C} | eq(\mathit{class}(c), \mathrm{Noun})\}$ – into Noun is either an object or a subject[5].

Figure 6.1 depicts a parsed example-phrase with pre-calculated concepts $\mathcal{C}$, which is used in the descriptions below.

Figure 6.1: Example for Dependency Grammar Analysis


Macro Dependency Graph:
VP
NP
NP
NP
PP
Phrase:
Wordclass:
Fragments:
Micro Dependency Graphs:
Set of concepts:
Planets are circling around the sun in elliptic orbits
noun verb verb adv det noun prep adj noun
NP VP NP NP
NP
NP
DP
NP
NP
AP
$C_{est}$ := { planet, sun, orbit }

### 6.2.1 Algorithm

**Determine the Node Order:** To get a an estimation of the Relations $\mathcal{R}_{estimation}$ we annotate the nodes (=: vertices $V$) of the Macro Dependency Graph $G = (V, E)$ in breath-first search order with ascending indices. With $Q$ := queue, $T$ := list of traversed nodes and $N_i(n)$ := the $i$-th neighbor of the current node $n$ we have:

```
$Q := T := \emptyset;$
$Q := Q \cup \{root(V)\};$
while ( $|Q| > 0$ ) :
    $n := Q_0; \quad Q := Q - \{n\};$                                    (6.11)
    $\forall_{i \in |N_i(n)| \wedge N_i(n) \notin T} : Q := Q \cup \{N_i(n)\};$
    $T := T + \{n\};$
end while;
```

The resulting List $T := \{T_0, \ldots T_{|T|}\}$ is finally a permutation of the set $V \in G$ in an appropriate order for the next step.

**Estimation-graph for Relations:** For each phrase $p$, an estimation graph $G_p$ is build:

$$G_p \subseteq O'' = (\mathcal{C}, \mathcal{R}) \tag{6.12}$$

$$G_p = (V, E) := (\mathcal{C}_{estimation,p}, \mathcal{R}_{estimation,p}) \tag{6.13}$$

[4] To get an approximate ontology $O$.

[5] For example $r := (c_1, c_2) :=$ “are circling around (planet,sun)” clarifies the role of subject and object by complete declaration of the verbal phrase VP; i.e. *planets circle around orbits* and not vice versa (:= *orbits circle around planets*).

The concepts $\mathcal{C}_{estimation}$ are given by the frequency analysis and its representation is each $c_i := lexeme(w)$. The set of relations $\mathcal{R}_{estimation}$ is derived from the following steps:

1. We start with a forest[6], solely consisting of the concepts $G_{p,0} = (\mathcal{C}_{estimation}, \emptyset)$.
2. The nodes are traversed in the above annotated order, i.e. we iterate over the list $T$ and define the current index as $i$ and thus the current node $t_i$:
   - If the current node $t_i$ is an verbal phrase, i.e. eq(fragmentType($t_i$),VP), we postulate an empty structure $r := text(t_i)\ (\emptyset, \emptyset)$ as a candidate for a relation $r \in \mathcal{R}_p$ ($\mathcal{R}_p$ is the set of relations for the current phrase $p$).
   - If the current node $t_i$ is an nominal phrase, i.e. eq(fragmentType($t_i$),NP) and lexeme($t_i$) $\in \mathcal{C}_{estimation}$, we extend the previous structure $r := r_{name}(c_1, c_2)$ in memory and set
     (a) in case eq$(c_1, \emptyset) \to c_1 := t_i$, or otherwise
     (b) in case eq$(c_2, \emptyset) \to c_2 := t_i$.

     Case (b) accomplishes the relation $r$ and thus it is added to the phrase relations $\mathcal{R}_p$. The graph is than updated to:

$$G_{p,i+1} := (\mathcal{C}_{estimation} \in G_{p,i},\ \mathcal{R}_p \in G_{p,i}\ \cup\ \{r\}) \tag{6.14}$$

Since each $c_i$ is a lexeme, we loose information, if the used word in the phrase for $c_i$ is e.g. in plural form. So, we can further annotate this to the relation (lhs := left hand side := $c_1$, rhs := right hand side := $c_2$):

$$annotate(r) := (lexeme(lhs) < lexemetype >) \mid (lexeme(rhs) < lexemetype >) \tag{6.15}$$

While adding $c_i =:$ NP to a relation, the <u>micro</u> dependency grammar can add attributes; e.g. in form of adjectives. This upgrades the set $\mathcal{C}$ of subjects:

$$annotate(c_i) := \text{Adjective}(w_i \in G_{p,micro}(c_i)) \tag{6.16}$$

Then $c_i :=$ NP is attributed to $attr(\text{Noun}) := \{Adj\}$.

### 6.2.2 Example

Starting from the phrase in figure 6.1 we sort the Macro Dependency Graph in breath first search order; according to equation 6.11:

1. VP(are circling around)
2. NP(planets) $\to_{micro}$ $\vdash$ N(planet<plural>) $\dashv$
3. NP(the sun) $\to_{micro}$ $\vdash$ Noun(sun) — Det(the) $\dashv$
4. NP(elliptic orbits) $\to_{micro}$ $\vdash$ N(orbits<plural>) — Adj(elliptic) $\dashv$
5. PP(in)

The first estimation graph $G_0$ equals (with concepts from word frequency analysis) to:

$$G_{p,0} = (\mathcal{C}_{estimation}, \emptyset) := (\{\text{planet}, \text{sun}, \text{orbit}\}, \emptyset) \tag{6.17}$$

Application of the algorithm results in tabular form. "are circling around" is abbreviated to "a.c.a.":

[6] A *forest* is a directed graph with a number of incoherent components greater than (or equal to) one and each is a tree.

Table 6.2: Example Extraction of Relations $\mathcal{R}$

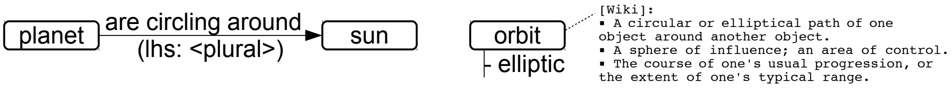
| **Step** | **Meta-token** | **Type** | $\in \mathcal{C}_{\text{est.}}$ | **Relation** $r$ | **Graph** |
|---|---|---|---|---|---|
| 1 | a.c.a. | VP | - | $r_1 := \text{a.c.a.}(\emptyset, \emptyset)$ | $G_{p,0}$ |
| 2 | planets | NP | true | $r_1 := \text{a.c.a.}(\emptyset, \emptyset)$ | $G_{p,0}$ |
| 2.1 | *lexeme*(planets) := planet | N | true | $r_1 := \text{a.c.a.}(\text{planet} < \textit{plural} >, \emptyset)$ | $G_{p,0}$ |
| 3 | the sun | NP | true | $r_1 := \text{a.c.a.}(\text{planet} < \textit{plural} >, \emptyset)$ | $G_{p,0}$ |
| 3.1 | sun | N | true | $r_1 := \text{a.c.a.}(\text{planet} < \textit{plural} >, \text{sun})$ | $G_{p,0}$ |
| 3.2 | the | DET | - | $r_1 := \text{a.c.a.}(\text{planet} < \textit{plural} >, \text{sun})$ | $G_{p,1}$ |
| 4 | elliptic orbits | NP | true | $r_2 := \emptyset(\emptyset, \emptyset)$ | $G_{p,1}$ |
| 4.1 | *lexeme*(orbits) := orbit | N | true | $r_2 := \emptyset(\text{planet<plural>}, \emptyset)$ | $G_{p,1}$ |
| 4.2 | elliptic | ADJ | - | $r_2 := \emptyset(c_1 := \text{orbit} < \textit{plural} >, \emptyset)$,<br>$\text{Adj}(c_1) := \text{elliptic}$ | $G_{p,1}$ |
| 5 | in | PP | - | $r_2 := \emptyset(c_1 := \text{orbit} < \textit{plural} >, \emptyset)$<br>$\text{Adj}(c_1) := \text{elliptic}$ | $G_{p,1}$ |

The final result for phrase $p$ is the partial ontology $O''_p = (\mathcal{C}_p, \mathcal{R}_p, \emptyset, \emptyset)$. The diagram is depicted in 6.2; and additionally shows semantics from the Wiktionary.

$$\mathcal{C}_p := \mathcal{C}_p \in G_{p,1} := \{\text{planet}, \text{sun}, \text{orbit} : \textit{Adj}(\{\text{ellptic}\})\} \tag{6.18}$$

$$\mathcal{R}_p := \mathcal{R}_p \in G_{p,1} := \{r_1 := \text{VP}(c_i, c_j) := \text{a.c.a.}(\text{planet} < \textit{plural} >, \text{sun})\} \tag{6.19}$$

Figure 6.2: Diagram for Graph $G_p, 1$


planet
are circling around
(lhs: <plural>)
sun
orbit
elliptic
[Wiki]:
▪ A circular or elliptical path of one object around another object.
▪ A sphere of influence; an area of control.
▪ The course of one's usual progression, or the extent of one's typical range.

## 6.3 Hyperonomy Analysis (Hierarchy of Concepts $\mathcal{HC}$)

Since the preprocessing implementation provides the partial[7] access to the set of semantic relations for a word $w$, we extend the ontological knowledge to a hierarchy of concepts, by evaluation of hyperonyms for every concept $c \in \mathcal{C}$. The set of concepts $\mathcal{C}$ is given by:

$$\mathcal{C}_{estimation} \ := \ \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\} \tag{6.20}$$

For each concept $c_i$ the lexical semantic net *WordNet* deliver the set of hyperonyms *hyp*:

$$hyp(c_i) \ := \ \{hc_1, hc_2, \ldots, hc_n\} \tag{6.21}$$

A single $hc_i$ is expressed by:

$$hc_i \ := < hyperonym, concept > := < c_k, c_i > \tag{6.22}$$

[7] Only synonyms and hyperonyms are extracted to keep simplicity.

The following simple algorithm with run-time $\mathcal{O}(\underbrace{\overline{hyp(c_i)}}_{\text{const}} \cdot n^2) \in \mathcal{O}(n^2)$, $(n := |\mathcal{C}|)$ gathers the hierarchy of concepts $\mathcal{HC}_{estimation}$:

$$\begin{array}{ll} (1.) & \mathcal{HC}_{est} \ := \ \emptyset \\ (2.) & \forall_{c_i \in \mathcal{C}_{est}} \ : \ \forall_{hc_j \in hyp(c_i)} \ : \ \forall c_k \in \mathcal{C}_{est} \setminus \{c_i\} \ : \\ & \quad match(hc_j, c_k) \ \rightarrow \ \mathcal{HC}_{est} \ := \ \mathcal{HC}_{est} \cup \{< c_k, c_i >\} \end{array} \tag{6.23}$$

Since $hyp(c_i)$ only delivers *words*, it must be matched with existing concepts:

$$match(hc_j, c_k) \ := \ eq(hc_j, c_k) \ \vee \ \exists_{w \in hc_j} \ : \ eq(w, c_k) \tag{6.24}$$

We furthermore demand that the *last* word must match, if the hyperonym is described with more than one word. An example is:

$$hyp(\text{lepton}) := \{hc_0 := \text{elementary particle}, \ hc_1 := \text{fundemental particle}\}$$

The relevant and last word is each particle. We assume that WordNet always has the form:

$$\begin{array}{rrcl} & hc_j & := & [ \ \ eq(class(hc_{j,0}), Noun) \ \ ] \\ \text{or} & hc_j & := & [ \ \ eq(class(hc_{j,0}), Adjective) \ \ , \ \ eq(class(hc_{j,1}), Noun) \ \ ] \end{array} \tag{6.25}$$

### 6.3.1 Example

$$\mathcal{C}_{est} := \{\text{planet}, \text{sun}, \text{star}, \text{heavenly body}, \text{galaxy}, \dots\} \tag{6.26}$$

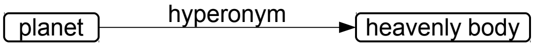
$$\begin{array}{ll} hyp_{WordNet}(\text{sun}) & := \{\text{star}\} \\ hyp_{WordNet}(\text{planet}) & := \{\text{heavily body}, \text{celestial body}, \text{follower}\} \end{array} \tag{6.27}$$

$$\mathcal{C}_{est} \cap hyp_{WordNet}(\text{planet}) \ := \ \{\text{heavily body}\} \tag{6.28}$$

Thus we get the set $\mathcal{HC}_{est}$:

$$\mathcal{HC}_{est} := \{< \text{heavenly body}, \text{planet} >, \ \dots\} \tag{6.29}$$

Figure 6.3: Partial Diagram for $\mathcal{HC}_{est}$



## 6.4 Consolidation and Ontology Synthesis

### 6.4.1 Unification of Partial Ontologies

The previous sections described the process of constructing the mathematical objects of the quadruple for the final ontology:

$$O' = ( \ \mathcal{C} := \{c_1, c_2, \dots\}, \ \mathcal{R} := \{R_{est,p1}, R_{est,p2}, \dots\}, \ \mathcal{HC} := \{hc_1, hc_2, \ \dots\}, \emptyset \ ) \tag{6.30}$$

While the Concepts $\mathcal{C}$ and Hierarchies of Concepts $\mathcal{HC}$ were synthesized for the total corpus, this is *not* true for the set or Relations $\mathcal{R}$. The latter were build separately for each phrase $p$. In equation 6.13 we built a phrasal relation-graph $G_p$. We denoted:

$$G_p = (V, E) := ( \ \mathcal{C}_{estimation,p}, \ \mathcal{R}_{estimation,p} \ )$$

Consolidation of the relations means unification of all phrasal sub graphs $G_p$:

$$G := \cup_{p \ \in \ texts} \ : \ G_p \tag{6.31}$$

Finally, the relations are taken from the graphs set of edges: $\mathcal{R} := E \in G$. As far as the concepts in each sub graph $G_p$ are treated as references (and not copies) from the set $\mathcal{C}$; the vertices $V$ merge implicitly.

### 6.4.2 Ontology Synthesis in the Web Ontology Language

Since $O'$ is now complete, we may now export the systems internal representation to the interchangable format OWL := Web Ontology Language. OWL is based on RDF[8], RDFS[9] and XML[10]. The official reference can be found in [Owl04]. Listing 6.1 shows the prolog of an OWL-file and listing 6.2 the epilog:

Listing 6.1: Prolog of the OWL-file

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1204192502.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1204192502.owl">
  <owl:Ontology rdf:about=""/>
```

Listing 6.2: Epilog of the OWL-file

```
</rdf:RDF>
```

#### Concepts $\mathcal{C}$ and Hierarchies of Concepts $\mathcal{HC}$

For each concept $c_i \in \mathcal{C}$, we define an OWL-class like in the example in listing 6.3[11]:

Listing 6.3: Definition of a Concept in OWL

```
  <owl:Class rdf:ID="$\boxed{c_i \in \mathcal{C}}$">
  </owl:Class>
```

If a Hierarchy of Concept $hc_k \in \mathcal{HC}$ exists, with $hc_k :=< c_i, c_j >$, the OWL-classes have to be generated as in listing 6.4. An ordering of the set $\mathcal{HC}$, such that a reference is defined before its usage (here: definition of $c_i$ before $c_j$) is *not* necessary.

Listing 6.4: Definition of a Hierarchy of Concept in OWL

```
  <owl:Class rdf:ID="$\boxed{c_i \in \mathcal{C}}$">
  </owl:Class>

  <owl:Class rdf:ID="$\boxed{c_j \in \mathcal{C}}$">
    <rdfs:subClassOf rdf:resource="#$\boxed{c_i \in \mathcal{C}}$"/>
  </owl:Class>
```

[8] RDF := Resource Description Framework

[9] RDFS := RDF-Schema

[10] XML := Extensible Markup Language

[11] $c_i \in \mathcal{C}$ must be substituted by the name of the concept in format of a string.

### Relations $\mathcal{R}$

For each relation $r_k \in \mathcal{R}$, we define an OWL-Object Property like in the example in listing 6.5, if the relation is defined as $r_k := r_{k,name}(c_i, c_j)$:

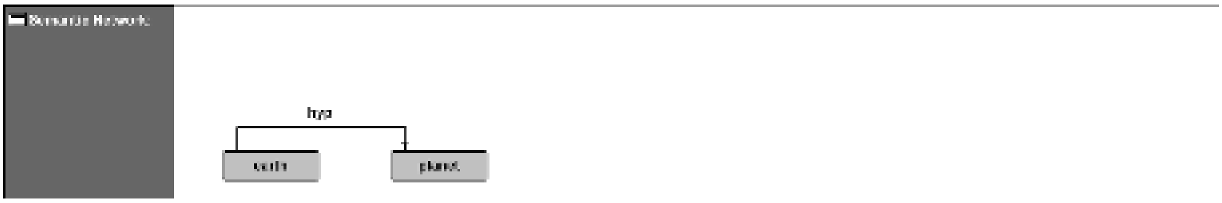
Listing 6.5: Definition of a Relation in OWL

```
<owl:ObjectProperty rdf:ID="$\boxed{r_k \in \mathcal{R}}$">
  <rdfs:domain rdf:resource="#$\boxed{c_i \in \mathcal{C}}$"/>
  <rdfs:range rdf:resource="#$\boxed{c_j \in \mathcal{C}}$"/>
</owl:ObjectProperty>
```

## 6.5 Example

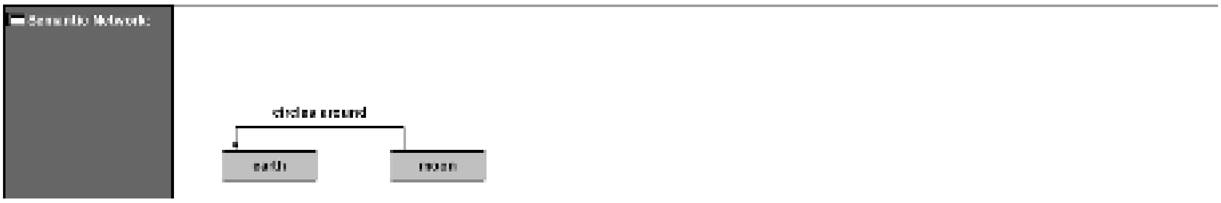
Figure 6.4 shows an example for the phrase-based extraction of Concepts and Hierarchy of Concepts for the example phrase “The earth is a planet”.

Figure 6.4: Example: $\mathcal{C}$ and $\mathcal{HC}$ for a Phrase



An example for an extracted relation is depicted in 6.5; for the example phrase “The moon circles around the earth in 28 days”.

Figure 6.5: Example: $\mathcal{R}$ for a Phrase



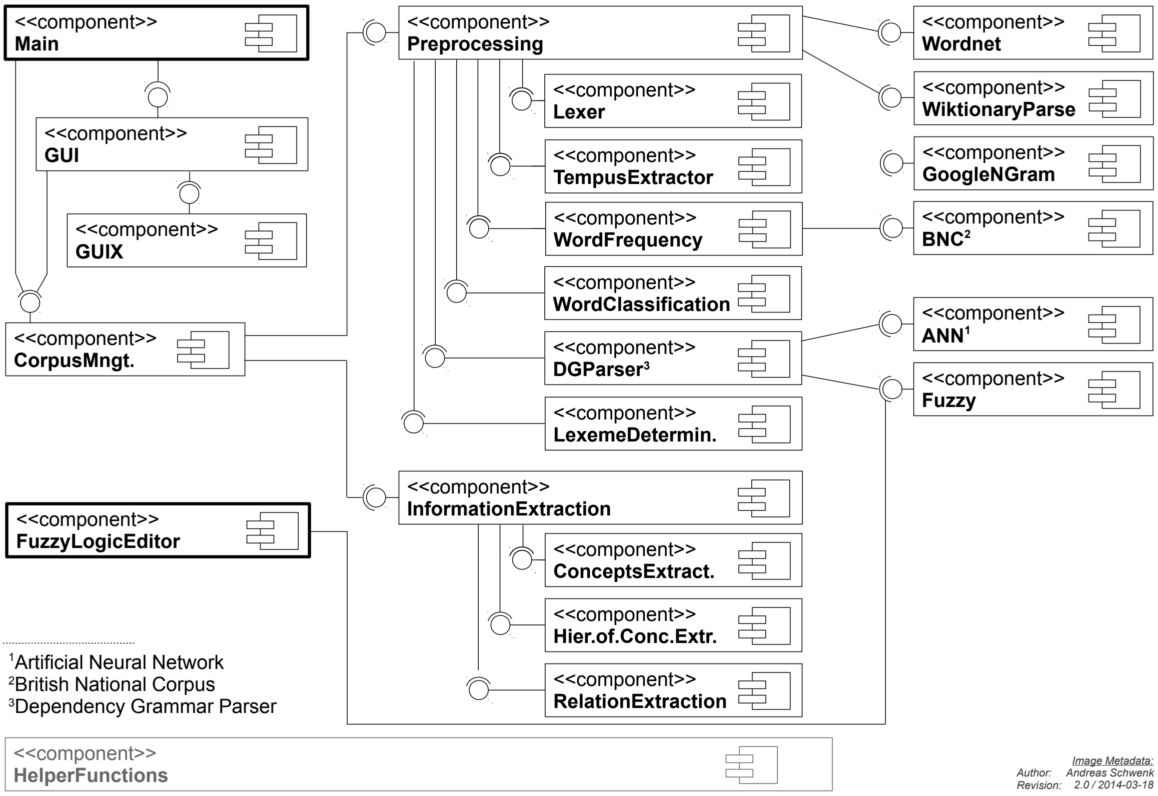
# Chapter 7

# Implementation Remarks

## 7.1 Components

The UML-Component Diagram 7.1 shows the real implementation. Since this project is *research* based, and the time is limited, all other information about the implementation may be taken only from the class diagrams (see below) and the complete source code (listed in the appendix).

Figure 7.1: UML Component Diagram



<<component>> Main
<<component>> GUI
<<component>> GUIX
<<component>> CorpusMngt.
<<component>> FuzzyLogicEditor
<<component>> Preprocessing
<<component>> Lexer
<<component>> TempusExtractor
<<component>> WordFrequency
<<component>> WordClassification
<<component>> DGParser³
<<component>> LexemeDetermin.
<<component>> InformationExtraction
<<component>> ConceptsExtract.
<<component>> Hier.of.Conc.Extr.
<<component>> RelationExtraction
<<component>> Wordnet
<<component>> WiktionaryParse
<<component>> GoogleNGram
<<component>> BNC²
<<component>> ANN¹
<<component>> Fuzzy
<<component>> HelperFunctions
¹Artificial Neural Network
²British National Corpus
³Dependency Grammar Parser
Image Metadata:
Author: Andreas Schwenk
Revision: 2.0 / 2014-03-18

## 7.2 Class Diagrams

Class diagrams are provided in the Appendix.

## 7.3 Statistics

The amount of source code is depicted in the following statistic 7.2. The y-axis measures the number of code lines; the x-axis shows the appropriate component:

Figure 7.2: Lines of Code =: LoC

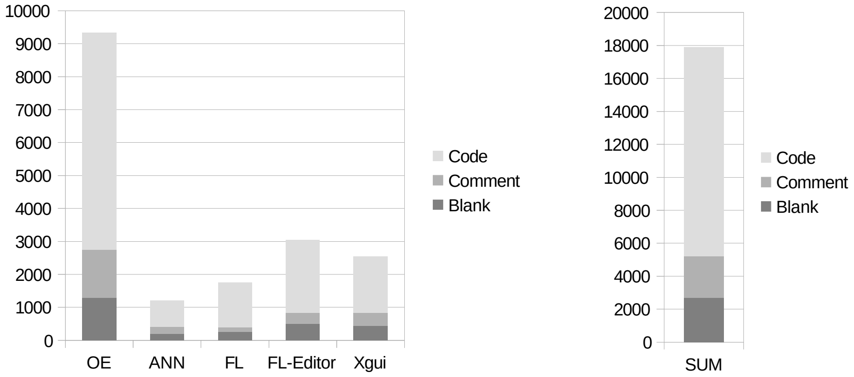


Table 7.1: Abbreviations of Software Components

| **Abbrev.** | **Description** |
|---|---|
| OE | Ontology Extraction |
| ANN | Artificial Neural Networks |
| FL | Fuzzy Logic |
| FL-Editor | Fuzzy Logic Editor |
| Xgui | Extended Graphical User Interface |

# Chapter 8

# Evaluation

As already stated in earlier chapters, we use the knowledge domain $D$ := “The Universe” as evaluation base. The concrete (partial) corpus is taken from Wikipedia and listed in table 8.1. To reduce the length of phrases, as well as the overall complexity, the “Simple English” variant of Wikipedia is used.

Table 8.1: Sources for the Knowledge Domain “The Universe”

| Title | Source |
|---|---|
| BigBang | `simple.wikipedia.org/wiki/BigBang` |
| Galaxy | `simple.wikipedia.org/wiki/Galaxy` |
| Sun | `simple.wikipedia.org/wiki/Sun` |
| Universe | `simple.wikipedia.org/wiki/Universe` |

## 8.1 Protégé

Protégé is “A free, open-source ontology editor and framework for building intelligent systems” [Pro]. The usage restricts here to visualization purposes, i.e. to show a synthesized ontology that is available in the OWL format (see chapter 6). Example screenshots for the class view (Concepts $\mathcal{C}$) and Hierarchies ($\mathcal{HC}$) are shown below:

Figure 8.1: Protege Class Hierarchy

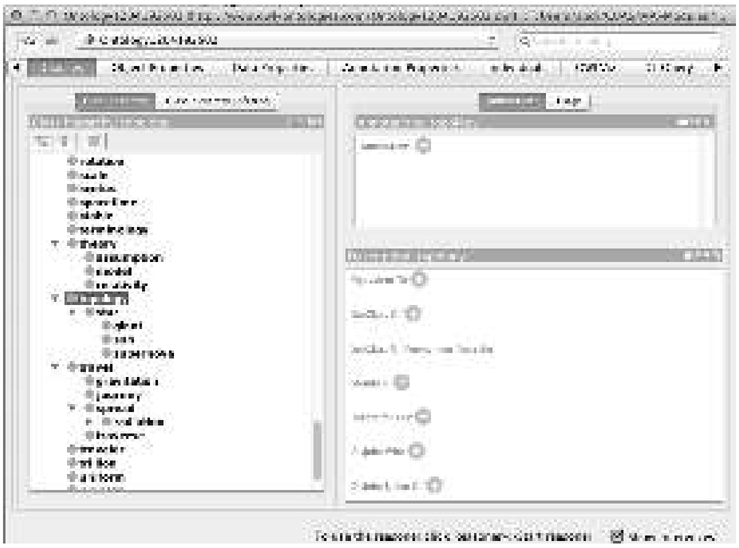
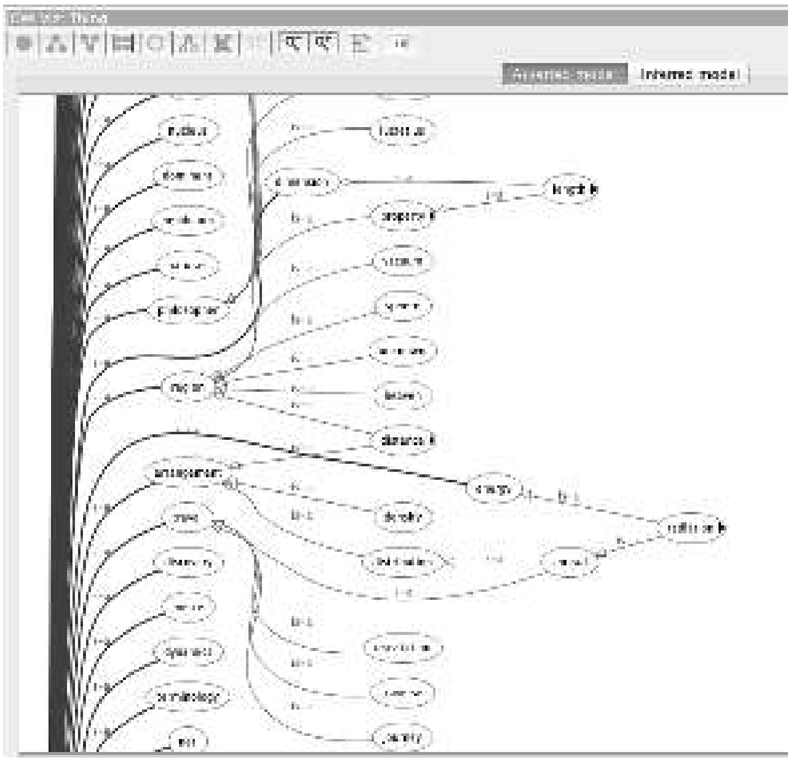


Figure 8.2: Protege OWL Viz



## 8.2 Resulting Ontology

The resulting ontology is shown in listing 8.1:

Listing 8.1: ontology.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns="http://www.owl-ontologies.com/Ontology1204192502.owl#"
  xml:base="http://www.owl-ontologies.com/Ontology1204192502.owl">
  <owl:Ontology rdf:about=""/>

  <owl:Class rdf:ID="lemaitre">
  </owl:Class>

  <owl:Class rdf:ID="galaxia">
  </owl:Class>

  <owl:Class rdf:ID="avicenna">
  </owl:Class>

  <owl:Class rdf:ID="lepton">
    <rdfs:subClassOf rdf:resource="#particle"/>
  </owl:Class>

  <owl:Class rdf:ID="redshift">
  </owl:Class>

  <owl:Class rdf:ID="baruch">
  </owl:Class>

  <owl:Class rdf:ID="lamda">
  </owl:Class>

  <owl:Class rdf:ID="antimatter">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="spacetime">
  </owl:Class>

  <owl:Class rdf:ID="sunspot">
  </owl:Class>

  <owl:Class rdf:ID="lowercase">
  </owl:Class>

  <owl:Class rdf:ID="univers">
  </owl:Class>

  <owl:Class rdf:ID="lucretius">
  </owl:Class>

  <owl:Class rdf:ID="graviton">
  </owl:Class>

  <owl:Class rdf:ID="photon">
  </owl:Class>

  <owl:Class rdf:ID="physic">
  </owl:Class>

  <owl:Class rdf:ID="spinoza">
  </owl:Class>

  <owl:Class rdf:ID="cosmology">
  </owl:Class>

  <owl:Class rdf:ID="planck">
    <rdfs:subClassOf rdf:resource="#physicist"/>
  </owl:Class>

  <owl:Class rdf:ID="ia">
  </owl:Class>

  <owl:Class rdf:ID="gravitation">
    <rdfs:subClassOf rdf:resource="#travel"/>
  </owl:Class>

  <owl:Class rdf:ID="topology">
  </owl:Class>

  <owl:Class rdf:ID="doesn">
  </owl:Class>

  <owl:Class rdf:ID="giordano">
  </owl:Class>

  <owl:Class rdf:ID="andromeda">
  </owl:Class>

  <owl:Class rdf:ID="supernova">
    <rdfs:subClassOf rdf:resource="#star"/>
  </owl:Class>

  <owl:Class rdf:ID="universe">
  </owl:Class>

  <owl:Class rdf:ID="helium">
    <rdfs:subClassOf rdf:resource="#element"/>
  </owl:Class>

  <owl:Class rdf:ID="johannes">
  </owl:Class>

  <owl:Class rdf:ID="deuterium">
    <rdfs:subClassOf rdf:resource="#atom"/>
  </owl:Class>

  <owl:Class rdf:ID="electromagnetism">
  </owl:Class>

  <owl:Class rdf:ID="wavelength">
    <rdfs:subClassOf rdf:resource="#distance"/>
  </owl:Class>

  <owl:Class rdf:ID="galaxy">
  </owl:Class>

  <owl:Class rdf:ID="trillion">
    <rdfs:subClassOf rdf:resource="#amount"/>
  </owl:Class>

  <owl:Class rdf:ID="datum">
  </owl:Class>

  <owl:Class rdf:ID="aristotle">
  </owl:Class>

  <owl:Class rdf:ID="atom">
  </owl:Class>

  <owl:Class rdf:ID="chlorophyll">
    <rdfs:subClassOf rdf:resource="#pigment"/>
  </owl:Class>

  <owl:Class rdf:ID="cicero">
  </owl:Class>

  <owl:Class rdf:ID="pythagoras">
  </owl:Class>

  <owl:Class rdf:ID="relativity">
```

```
    <rdfs:subClassOf rdf:resource="#theory"/>
  </owl:Class>

  <owl:Class rdf:ID="ellipse">
  </owl:Class>

  <owl:Class rdf:ID="copernicus">
  </owl:Class>

  <owl:Class rdf:ID="interact">
  </owl:Class>

  <owl:Class rdf:ID="nucleus">
  </owl:Class>

  <owl:Class rdf:ID="infra">
  </owl:Class>

  <owl:Class rdf:ID="epoch">
  </owl:Class>

  <owl:Class rdf:ID="metric">
    <rdfs:subClassOf rdf:resource="#measurement"/>
    <rdfs:subClassOf rdf:resource="#amount"/>
  </owl:Class>

  <owl:Class rdf:ID="hubble">
  </owl:Class>

  <owl:Class rdf:ID="expansion">
  </owl:Class>

  <owl:Class rdf:ID="newtonian">
  </owl:Class>

  <owl:Class rdf:ID="density">
    <rdfs:subClassOf rdf:resource="#arrangement"/>
  </owl:Class>

  <owl:Class rdf:ID="traverse">
    <rdfs:subClassOf rdf:resource="#travel"/>
  </owl:Class>

  <owl:Class rdf:ID="higgs">
  </owl:Class>

  <owl:Class rdf:ID="hoyle">
  </owl:Class>

  <owl:Class rdf:ID="undertone">
    <rdfs:subClassOf rdf:resource="#meaning"/>
  </owl:Class>

  <owl:Class rdf:ID="halo">
    <rdfs:subClassOf rdf:resource="#light"/>
  </owl:Class>

  <owl:Class rdf:ID="photosynthesis">
  </owl:Class>

  <owl:Class rdf:ID="eriugena">
  </owl:Class>

  <owl:Class rdf:ID="hydrogen">
    <rdfs:subClassOf rdf:resource="#element"/>
  </owl:Class>

  <owl:Class rdf:ID="momentum">
    <rdfs:subClassOf rdf:resource="#property"/>
  </owl:Class>

  <owl:Class rdf:ID="electron">
    <rdfs:subClassOf rdf:resource="#lepton"/>
  </owl:Class>

  <owl:Class rdf:ID="orbit">
  </owl:Class>

  <owl:Class rdf:ID="observation">
    <rdfs:subClassOf rdf:resource="#measurement"/>
  </owl:Class>

  <owl:Class rdf:ID="residue">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="sun">
    <rdfs:subClassOf rdf:resource="#star"/>
    <rdfs:subClassOf rdf:resource="#light"/>
    <rdfs:subClassOf rdf:resource="#radiation"/>
  </owl:Class>

  <owl:Class rdf:ID="imbalance">
    <rdfs:subClassOf rdf:resource="#property"/>
  </owl:Class>

  <owl:Class rdf:ID="chaucer">
  </owl:Class>

  <owl:Class rdf:ID="pendulum">
  </owl:Class>

  <owl:Class rdf:ID="pigment">
  </owl:Class>

  <owl:Class rdf:ID="gram">
  </owl:Class>

  <owl:Class rdf:ID="faster">
  </owl:Class>

  <owl:Class rdf:ID="equation">
  </owl:Class>

  <owl:Class rdf:ID="lithium">
    <rdfs:subClassOf rdf:resource="#element"/>
  </owl:Class>

  <owl:Class rdf:ID="eyesight">
  </owl:Class>

  <owl:Class rdf:ID="plasma">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="decrease">
  </owl:Class>

  <owl:Class rdf:ID="conservation">
    <rdfs:subClassOf rdf:resource="#principle"/>
  </owl:Class>

  <owl:Class rdf:ID="gravity">
  </owl:Class>

  <owl:Class rdf:ID="radiation">
    <rdfs:subClassOf rdf:resource="#energy"/>
  </owl:Class>

  <owl:Class rdf:ID="earth">
    <rdfs:subClassOf rdf:resource="#planet"/>
    <rdfs:subClassOf rdf:resource="#element"/>
  </owl:Class>

  <owl:Class rdf:ID="abundance">
    <rdfs:subClassOf rdf:resource="#ratio"/>
  </owl:Class>

  <owl:Class rdf:ID="quebec">
  </owl:Class>

  <owl:Class rdf:ID="diameter">
    <rdfs:subClassOf rdf:resource="#length"/>
  </owl:Class>

  <owl:Class rdf:ID="universum">
  </owl:Class>

  <owl:Class rdf:ID="definition">
  </owl:Class>

  <owl:Class rdf:ID="sphere">
    <rdfs:subClassOf rdf:resource="#region"/>
  </owl:Class>

  <owl:Class rdf:ID="doppler">
    <rdfs:subClassOf rdf:resource="#physicist"/>
  </owl:Class>

  <owl:Class rdf:ID="matter">
  </owl:Class>

  <owl:Class rdf:ID="jupiter">
    <rdfs:subClassOf rdf:resource="#planet"/>
    <rdfs:subClassOf rdf:resource="#giant"/>
  </owl:Class>

  <owl:Class rdf:ID="greek">
  </owl:Class>

  <owl:Class rdf:ID="terminology">
  </owl:Class>

  <owl:Class rdf:ID="astronomy">
  </owl:Class>

  <owl:Class rdf:ID="galileo">
    <rdfs:subClassOf rdf:resource="#astronomer"/>
  </owl:Class>

  <owl:Class rdf:ID="foucault">
    <rdfs:subClassOf rdf:resource="#physicist"/>
  </owl:Class>

  <owl:Class rdf:ID="mass">
    <rdfs:subClassOf rdf:resource="#property"/>
    <rdfs:subClassOf rdf:resource="#amount"/>
  </owl:Class>

  <owl:Class rdf:ID="energy">
  </owl:Class>

  <owl:Class rdf:ID="continuum">
  </owl:Class>

  <owl:Class rdf:ID="gauge">
  </owl:Class>

  <owl:Class rdf:ID="curvature">
    <rdfs:subClassOf rdf:resource="#form"/>
  </owl:Class>

  <owl:Class rdf:ID="prediction">
  </owl:Class>

  <owl:Class rdf:ID="metre">
  </owl:Class>

  <owl:Class rdf:ID="geometry">
  </owl:Class>

  <owl:Class rdf:ID="zero">
  </owl:Class>

  <owl:Class rdf:ID="dust">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="microwave">
    <rdfs:subClassOf rdf:resource="#radiation"/>
  </owl:Class>

  <owl:Class rdf:ID="net">
  </owl:Class>

  <owl:Class rdf:ID="observer">
  </owl:Class>

  <owl:Class rdf:ID="spiral">
    <rdfs:subClassOf rdf:resource="#rotation"/>
  </owl:Class>

  <owl:Class rdf:ID="measurement">
  </owl:Class>

  <owl:Class rdf:ID="particle">
  </owl:Class>

  <owl:Class rdf:ID="binoculars">
  </owl:Class>

  <owl:Class rdf:ID="correlation">
  </owl:Class>

  <owl:Class rdf:ID="cloud">
  </owl:Class>

  <owl:Class rdf:ID="lighter">
  </owl:Class>

  <owl:Class rdf:ID="vacuum">
    <rdfs:subClassOf rdf:resource="#region"/>
  </owl:Class>

  <owl:Class rdf:ID="planet">
  </owl:Class>

  <owl:Class rdf:ID="theory">
  </owl:Class>

  <owl:Class rdf:ID="fusion">
  </owl:Class>

  <owl:Class rdf:ID="stable">
  </owl:Class>

  <owl:Class rdf:ID="star">
    <rdfs:subClassOf rdf:resource="#topology"/>
  </owl:Class>

  <owl:Class rdf:ID="georges"></owl:Class>

  <owl:Class rdf:ID="light">
    <rdfs:subClassOf rdf:resource="#radiation"/>
    <rdfs:subClassOf rdf:resource="#property"/>
  </owl:Class>

  <owl:Class rdf:ID="preliminary"></owl:Class>

  <owl:Class rdf:ID="form">
    <rdfs:subClassOf rdf:resource="#property"/>
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="ours"></owl:Class>

  <owl:Class rdf:ID="moon">
    <rdfs:subClassOf rdf:resource="#light"/>
    <rdfs:subClassOf rdf:resource="#radiation"/>
  </owl:Class>

  <owl:Class rdf:ID="element"></owl:Class>

  <owl:Class rdf:ID="astronomer">
    <rdfs:subClassOf rdf:resource="#physicist"/>
  </owl:Class>

  <owl:Class rdf:ID="rotation"></owl:Class>

  <owl:Class rdf:ID="symmetry">
    <rdfs:subClassOf rdf:resource="#property"/>
  </owl:Class>

  <owl:Class rdf:ID="physicist">
    <rdfs:subClassOf rdf:resource="#scientist"/>
  </owl:Class>

  <owl:Class rdf:ID="traveler"></owl:Class>

  <owl:Class rdf:ID="ordinary"></owl:Class>

  <owl:Class rdf:ID="distance">
    <rdfs:subClassOf rdf:resource="#arrangement"/>
    <rdfs:subClassOf rdf:resource="#region"/>
  </owl:Class>

  <owl:Class rdf:ID="billion">
    <rdfs:subClassOf rdf:resource="#amount"/>
  </owl:Class>

  <owl:Class rdf:ID="creation"></owl:Class>

  <owl:Class rdf:ID="singularity"></owl:Class>

  <owl:Class rdf:ID="travel"></owl:Class>

  <owl:Class rdf:ID="speed">
    <rdfs:subClassOf rdf:resource="#ratio"/>
  </owl:Class>

  <owl:Class rdf:ID="everything"></owl:Class>

  <owl:Class rdf:ID="generations">
    <rdfs:subClassOf rdf:resource="#phase"/>
  </owl:Class>

  <owl:Class rdf:ID="albert"></owl:Class>

  <owl:Class rdf:ID="inflation">
    <rdfs:subClassOf rdf:resource="#explosion"/>
    <rdfs:subClassOf rdf:resource="#expansion"/>
  </owl:Class>

  <owl:Class rdf:ID="dimension"></owl:Class>

  <owl:Class rdf:ID="explosion"></owl:Class>

  <owl:Class rdf:ID="distribution">
    <rdfs:subClassOf rdf:resource="#arrangement"/>
  </owl:Class>

  <owl:Class rdf:ID="region">
  </owl:Class>

  <owl:Class rdf:ID="uniform">
  </owl:Class>

  <owl:Class rdf:ID="grid">
    <rdfs:subClassOf rdf:resource="#form"/>
  </owl:Class>

  <owl:Class rdf:ID="invention">
    <rdfs:subClassOf rdf:resource="#creation"/>
  </owl:Class>

  <owl:Class rdf:ID="ratio">
  </owl:Class>

  <owl:Class rdf:ID="chemical"></owl:Class>

  <owl:Class rdf:ID="limitation"></owl:Class>

  <owl:Class rdf:ID="unknown">
    <rdfs:subClassOf rdf:resource="#region"/>
  </owl:Class>

  <owl:Class rdf:ID="lifetime"></owl:Class>

  <owl:Class rdf:ID="overall"></owl:Class>

  <owl:Class rdf:ID="whereas"></owl:Class>

  <owl:Class rdf:ID="giant">
    <rdfs:subClassOf rdf:resource="#star"/>
  </owl:Class>

  <owl:Class rdf:ID="length">
    <rdfs:subClassOf rdf:resource="#property"/>
    <rdfs:subClassOf rdf:resource="#dimension"/>
  </owl:Class>

  <owl:Class rdf:ID="bubble"></owl:Class>

  <owl:Class rdf:ID="ancient"></owl:Class>

  <owl:Class rdf:ID="netherlands"></owl:Class>

  <owl:Class rdf:ID="assumption">
    <rdfs:subClassOf rdf:resource="#theory"/>
  </owl:Class>

  <owl:Class rdf:ID="background"></owl:Class>

  <owl:Class rdf:ID="kg"></owl:Class>

  <owl:Class rdf:ID="accuracy"></owl:Class>

  <owl:Class rdf:ID="dynamics"></owl:Class>

  <owl:Class rdf:ID="amount"></owl:Class>

  <owl:Class rdf:ID="arrangement"></owl:Class>

  <owl:Class rdf:ID="meaning">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:Class rdf:ID="worship"></owl:Class>

  <owl:Class rdf:ID="electrical"></owl:Class>

  <owl:Class rdf:ID="average"></owl:Class>

  <owl:Class rdf:ID="principle"></owl:Class>

  <owl:Class rdf:ID="scientist"></owl:Class>

  <owl:Class rdf:ID="outer"></owl:Class>

  <owl:Class rdf:ID="twist">
    <rdfs:subClassOf rdf:resource="#rotation"/>
  </owl:Class>

  <owl:Class rdf:ID="existence"></owl:Class>

  <owl:Class rdf:ID="property">
    <rdfs:subClassOf rdf:resource="#region"/>
  </owl:Class>

  <owl:Class rdf:ID="phase">
    <rdfs:subClassOf rdf:resource="#matter"/>
  </owl:Class>

  <owl:ObjectProperty rdf:ID="SYNONYM">
    <rdfs:domain rdf:resource="#sphere"/>
    <rdfs:range rdf:resource="#orbit"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="SYNONYM">
    <rdfs:domain rdf:resource="#dimension"/>
    <rdfs:range rdf:resource="#property"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#earth"/>
    <rdfs:range rdf:resource="#planet"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="circles␣around">
    <rdfs:domain rdf:resource="#moon"/>
    <rdfs:range rdf:resource="#earth"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#theory"/>
    <rdfs:range rdf:resource="#universe"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="proved">
    <rdfs:domain rdf:resource="#redshift"/>
    <rdfs:range rdf:resource="#universe"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="was">
    <rdfs:domain rdf:resource="#observation"/>
    <rdfs:range rdf:resource="#singularity"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is␣known">
    <rdfs:domain rdf:resource="#universe"/>
    <rdfs:range rdf:resource="#radiation"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="takes␣place">
    <rdfs:domain rdf:resource="#expansion"/>
    <rdfs:range rdf:resource="#universe"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#mass"/>
    <rdfs:range rdf:resource="#galaxy"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is␣produced">
    <rdfs:domain rdf:resource="#gravitation"/>
    <rdfs:range rdf:resource="#matter"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#energy"/>
    <rdfs:range rdf:resource="#star"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#energy"/>
    <rdfs:range rdf:resource="#earth"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="was">
    <rdfs:domain rdf:resource="#cloud"/>
    <rdfs:range rdf:resource="#light"/>
  </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="is">
    <rdfs:domain rdf:resource="#energy"/>
    <rdfs:range rdf:resource="#earth"/>
  </owl:ObjectProperty>

</rdf:RDF>
```

# Chapter 9

# Conclusion

The general project goal could be reached: A knowledge acquisition system was designed, implemented and evaluated. The system is capable of extracting partial ontological information. As expected from a natural language processing system, the uncertainty could not converge to zero. The quality of the set of concepts $\mathcal{C}$ is acceptable, i.e. one can observe a high correlation of subjects from the testing domain $D :=$ "The universe". Hierarchy of Concepts $\mathcal{HC}$ can be rated to be even more accurate. The quality of Relations $\mathcal{R}$ is not as high as expected: Considering the dependency grammar is a reliable basis, but the relation extraction itself could be refined.

Despite the information extraction, a huge amount of time was spend to the preprocessing part, which consists of a syntactic and a semantic analysis, enrichment of worth with semantic relations, the determination of the word classes and word frequencies as well as a prediction of the used tense.

A new Neuro-Fuzzy based approach for hierarchical parsing dependency grammar could be presented and implemented; the quality depends on the user learning process and can be iteratively improved.

The focus on own implementations, instead of using external libraries (a) helped for a deep understanding (b) showed that sufficient implementations must not always rely on long evaluated external projects and (c) even advanced to new approaches, e.g. a Neuro-Fuzzy system for dependency grammar parsing. Detailed critical evaluations for specialized topics can be found in most chapters.

# Bibliography

[AG05] Khurshid Ahmad and Lee Gillam. *Automatic Ontology Extraction from Unstructured Texts.* University of Surrey `https://www.scss.tcd.ie/Khurshid.Ahmad/Research/OntoTerminology/ODBASE2005.final.pdf`, 2005.

[Bar13] Prof. Rainer Bartz. *Computational Intelligence – Lecture.* Cologne University of Applied Sciences. `http://www.nt-rt.fh-koeln.de/index.html`, 2013.

[Bri14] *About the British National Corpus.* `http://www.natcorp.ox.ac.uk/corpus/index.xml`, 2014.

[CFL12] Alexander Clark, Chris Fox, and Shalom Lappin. *The Handbook of Computational Linguistics and Natural Language Processing.* John Wiley & Sons. ISBN 978-1-118-34718-8, 2012.

[CM02] Andrew Carstairs-McCarthy. *An Introduction to English Morphology: Words and Their Structure.* Edinburgh University Press. ISBN 0 7486 1326 9, 2002.

[Cov01] Michael A. Covington. *A Fundamental Algorithm for Dependency Parsing.* The University of Georgia `http://www.stanford.edu/~mjkay/covington.pdf`, 2001.

[DOM] *XML DOM Parser.* W3 Schools `http://www.w3schools.com/dom/dom_parser.asp`.

[EP08] Katrin Erk and Prof. Lutz Priese. *Theoretische Informatik.* Springer. ISBN 978-3-540-76319-2, 2008.

[Gai65] H. Gaifman. *Dependency Systems and Phrase Structure Systems.* Inf. Control, 1965.

[Irr14] *List of English Irregular Verbs.* `http://www.usingenglish.com/reference/irregular-verbs/`, 2014.

[Joh65] Elmer D. Johnson. *A history of libraries in the western world.* Scarecrow Press, New Jersey. ISBN 0-8108-0949-4, 1965.

[LGT96] Steve Lawrence, C. Lee Giles, and Ah Chung Tsoi. *What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation.* Pennsylvania State University `http://clgiles.ist.psu.edu/papers/UMD-CS-TR-3617.what.size.neural.net.to.use.pdf`, 1996.

[Lin14] *Link Grammar.* `http://www.link.cs.cmu.edu/link/`, 2014.

[MV01] Alexander Maedche and Raphael Volz. *The Ontology Extraction & Maintenance Framework Text-To-Onto.* `http://users.csc.calpoly.edu/~fkurfess/Events/DM-KM-01/Volz.pdf`, 2001.

[Owl04] *OWL Web Ontology Language – Semantics and Abstract Syntax.* `http://www.w3.org/TR/owl-semantics/`, 2004.

[Pro] *Protégé.* Stanford University. `http://protege.stanford.edu`.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors.* Nature – International Weekly Journal of Science, 1986.

[RN10] Studart Russell and Peter Norvig. *Artificial Intelligence – A Modern Approach.* Pearson Education. ISBN 978-0-13-207148-2, 2010.

[SS92] John F. Sowa and Stuart C. Shapiro. *Encyclopedia of Artificial Intelligence.* Wiley. ISBN 978-0471503071 `http://www.jfsowa.com/pubs/semnet.htm`, 1992.

[Stu] *Study sheet for semantics.* `http://pandora.cii.wwu.edu/vajda/ling201/test3materials/semanticsHANDOUT.htm`.

[Tem] *Verb Tenses.* University of Washington. `http://faculty.washington.edu/marynell/grammar/verbtenses.html`.

[Wik14a] *Information Explosion.* Wikipedia `http://en.wikipedia.org/wiki/Information_explosion`, 2014.

[Wik14b] *Treebank.* Wikipedia `http://en.wikipedia.org/wiki/Treebank`, 2014.

[Wir96] Niklaus Wirth. *Compiler Construction.* Addison-Wesley. ISBN 978-0201403534, 1996.

[Wor14] *Wordnet.* `http://wordnet.princeton.edu`, 2014.

# Appendices

# Appendix A

# Document Type Definitions

The following listings show the Document Type Definitions (=: DTDs) to specify the XML files, that are used in the project:

Listing A.1: fuzzy.dtd

```
<!ELEMENT fuzzy (inputs, outputs, inference, defuzzification)>
<!ELEMENT inputs (LV*)>
<!ELEMENT outputs (LV*)>
<!ELEMENT LV (LT*, rendering)>
<!ATTLIST LV
    name         CDATA #REQUIRED
>
<!ATTLIST LT
    name         CDATA #REQUIRED
    type         CDATA #REQUIRED
    xl           CDATA #IMPLIED
    xr           CDATA #IMPLIED
    xm           CDATA #IMPLIED
    xm1          CDATA #IMPLIED
    xm2          CDATA #IMPLIED
    a            CDATA #IMPLIED
    b            CDATA #IMPLIED
    sigma        CDATA #IMPLIED
>
<!ATTLIST rendering
    left         CDATA #REQUIRED
    right        CDATA #REQUIRED
    major        CDATA #REQUIRED
    minor        CDATA #REQUIRED
>
<!ELEMENT inference (rule*)>
<!ATTLIST inference
    aggregation  CDATA #REQUIRED
    accumulation CDATA #REQUIRED
    activation   CDATA #REQUIRED
>
<!ELEMENT rule (#PCDATA)>
<!ATTLIST rule
    name         CDATA #REQUIRED
>
<!ATTLIST defuzzification
    mathod       CDATA #REQUIRED
>
```

Listing A.2: fuzzyrules.dtd

```
<!ELEMENT fuzzyrules (rule*)>
<!ELEMENT rule (#PCDATA)>
<!ATTLIST rule
    name         CDATA #REQUIRED
>
```

Listing A.3: article.dtd

```
<!ELEMENT article (name, chapter)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT chapter (phrase*, chapter*)>
<!ATTLIST chapter
    title      CDATA #REQUIRED
    id         CDATA #REQUIRED
>
<!ELEMENT phrase (#PCDATA)>
<!ATTLIST phrase
    id         CDATA #REQUIRED
>
```

Listing A.4: preprocessing.dtd

```
<!ELEMENT preprocessing (phrase)>

<!ELEMENT phrase (tempus, syntax, dependency, annotations)>
<!ATTLIST phrase
    id           CDATA #REQUIRED
    text         CDATA #REQUIRED
>

<!ATTLIST tempus
    t            CDATA #REQUIRED
>

<!ELEMENT syntax (fragment|word|greek_word|number|symbol|period|END|Comma|Semicolon|
    OpeningParenthesis|ClosingParenthesis|Plus|Minus|Percent|Quotes)*>
<!ATTLIST fragment
    type         CDATA #REQUIRED
    size         CDATA #REQUIRED
>
<!ELEMENT word (#PCDATA)>
<!ATTLIST word
    class        CDATA #REQUIRED
    lexeme       CDATA #REQUIRED
    lexemeType   CDATA #REQUIRED
    idx          CDATA #REQUIRED
>
<!ELEMENT greek_word (#PCDATA)>
<!ATTLIST greek_word
    idx          CDATA #REQUIRED
>
<!ELEMENT number (#PCDATA)>
<!ATTLIST number
    idx          CDATA #REQUIRED
>
<!ELEMENT symbol (#PCDATA)>
<!ATTLIST symbol
    idx          CDATA #REQUIRED
>

##### same for every other token type #####

<!ATTLIST period
    idx          CDATA #REQUIRED
>
<!ATTLIST end
    idx          CDATA #REQUIRED
>

<!ELEMENT dependency (Verb)>

<!ELEMENT Verb (Verb|Noun|Adjective|Adverb|Preposition|Determinative|Conjunction|Interjection|
    Numeral|UNKNOWN|Comma|Period)*>
<!ATTLIST Verb
    word         CDATA #REQUIRED
    idx          CDATA #REQUIRED
    fuzzy        CDATA #REQUIRED
>
<!ELEMENT Noun (Verb|Noun|Adjective|Adverb|Preposition|Determinative|Conjunction|Interjection|
    Numeral|UNKNOWN|Comma|Period)*>
<!ATTLIST Noun
    word         CDATA #REQUIRED
    idx          CDATA #REQUIRED
    fuzzy        CDATA #REQUIRED
>

##### ... same for every other word class ... #####

<!ELEMENT annotations (word*)>
<!ELEMENT word (wordfrequency,semantics,synonyms,hyperonyms)>
<!ATTLIST word
    w            CDATA #REQUIRED
    idx          CDATA #REQUIRED
>
<!ATTLIST wordfrequency
    word         CDATA #REQUIRED
    lexeme      CDATA #REQUIRED
>
<!ELEMENT semantics (class*)>
<!ELEMENT class (semantic*)>
<!ATTLIST class
    type        CDATA #REQUIRED
    probability CDATA #REQUIRED
>
<!ELEMENT semantic (#PCDATA)>
<!ATTLIST semantic
    sid         CDATA #REQUIRED
>
<!ELEMENT synonyms (synonym*)>
<!ATTLIST synonym
    sid         CDATA #REQUIRED
    word        CDATA #REQUIRED
>
<!ELEMENT hyperonyms (hyperonym*)>
<!ATTLIST hyperonym
    hid         CDATA #REQUIRED
    word        CDATA #REQUIRED
>
```

Listing A.5: TrainingSets.dtd

```
<!ELEMENT TrainingSets (TS*)>
<!ELEMENT TS (Type, Phrase, WordClasses, Data, TimeStamp)>
<!ATTLIST TS
    index       CDATA #REQUIRED
>
<!ATTLIST Type
    t           CDATA #REQUIRED
>
<!ATTLIST Phrase
    text        CDATA #REQUIRED
>
<!ATTLIST WordClasses
    wc          CDATA #REQUIRED
>
<!ATTLIST Data
    parentIndex CDATA #REQUIRED
    parentName  CDATA #REQUIRED
    childIndex  CDATA #REQUIRED
    childName   CDATA #REQUIRED
    link        CDATA #REQUIRED
>
<!ATTLIST TimeStamp
    ts          CDATA #REQUIRED
>
```

# Appendix B

# Fuzzy Logic Editor

## B.1 Definition of a FLS via an XML File

Listing B.1: Dependency Grammar FLS Properties: Excerpt

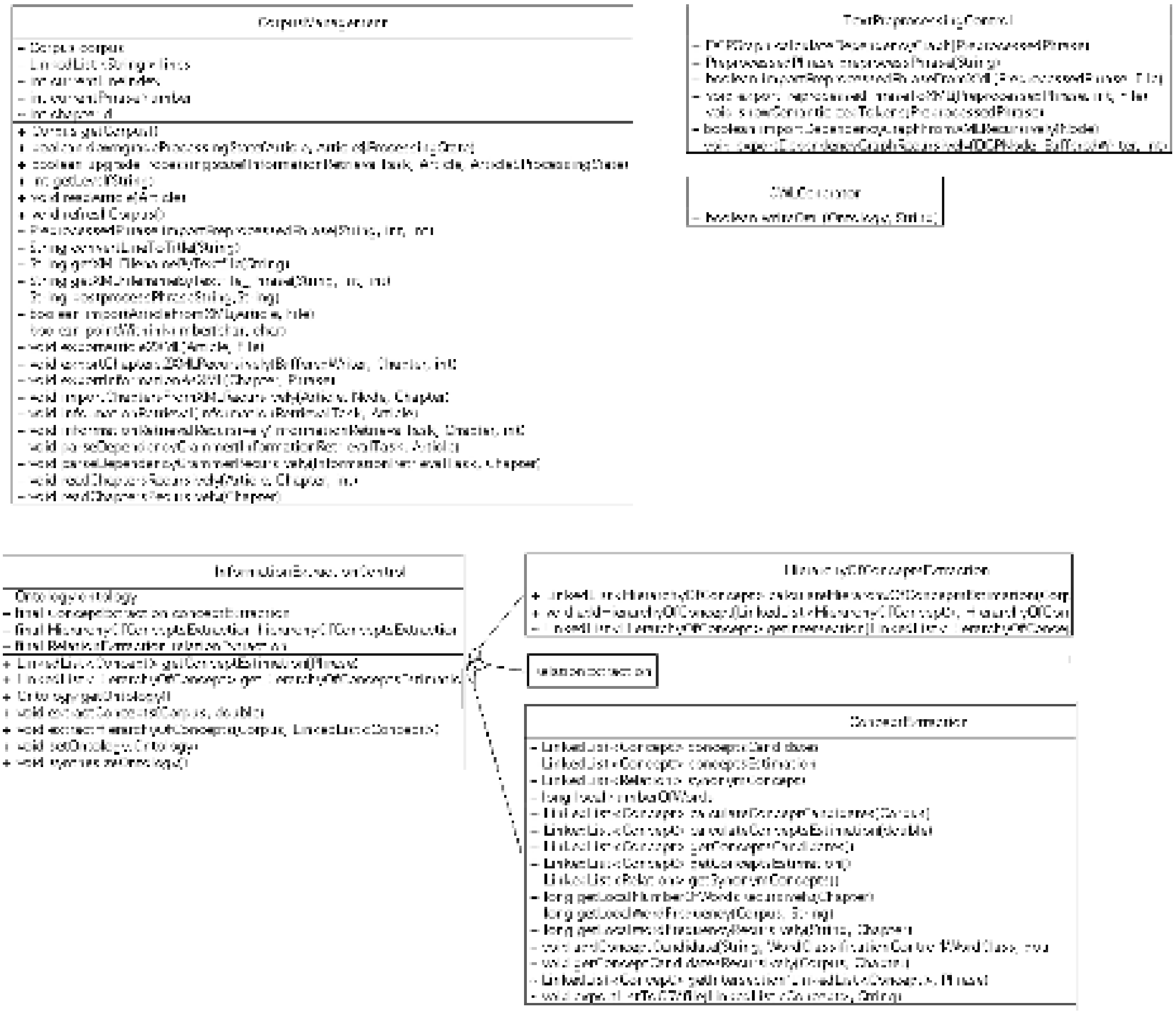
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE fuzzy SYSTEM "fuzzy">

<fuzzy>
  <inputs>
    <LV name="position">
      <LT name="positiveSmall" type="Trapezoidal" xl="-1.0" xm1="-1.0" xm2="5.0" xr="15.0"/>
      <LT name="positiveLarge" type="Trapezoidal" xl="0.0" xm1="15.0" xm2="40.0" xr="100.0"/>
      <rendering left="0.0" right="50.0" major="5.0" minor="1.0"/>
    </LV>
    <LV name="relativePosition">
      <LT name="negativeSmall" type="Trapezoidal" xl="-5.0" xm1="-2.0" xm2="-1.0" xr="0.0"/>
      <LT name="positiveSmall" type="Trapezoidal" xl="0.0" xm1="1.0" xm2="2.0" xr="5.0"/>
      <LT name="zero" type="Triangular" xl="-1.0" xm="0.0" xr="1.0"/>
      <LT name="positiveLarge" type="RightHandSaddle" xl="3.0" xr="7.0"/>
      <LT name="negativeLarge" type="LeftHandSaddle" xl="-7.0" xr="-3.0"/>
      <LT name="positive" type="Trapezoidal" xl="0.0" xm1="1.0" xm2="75.0" xr="100.0"/>
      <rendering left="-10.0" right="10.0" major="5.0" minor="1.0"/>
    </LV>
    <LV name="root">
      <LT name="false" type="LeftHandSaddle" xl="0.1" xr="0.3"/>
      <LT name="true" type="RightHandSaddle" xl="0.7" xr="0.9"/>
      <rendering left="0.0" right="1.0" major="0.2" minor="0.1"/>
    </LV>
    <LV name="noun">
      <LT name="low" type="Sigmoid" a="10.0" b="0.3"/>
      <LT name="high" type="Sigmoid" a="-10.0" b="0.7"/>
      <rendering left="0.0" right="1.0" major="0.1" minor="0.1"/>
    </LV>
    ...
    <LV name="nounParent">
      <LT name="low" type="Sigmoid" a="10.0" b="0.3"/>
      <LT name="high" type="Sigmoid" a="-10.0" b="0.7"/>
      <rendering left="0.0" right="1.0" major="0.1" minor="0.1"/>
    </LV>
    <LV name="verbParent">
      <LT name="low" type="Sigmoid" a="10.0" b="0.3"/>
      <LT name="high" type="Sigmoid" a="-10.0" b="0.7"/>
      <rendering left="0.0" right="1.0" major="0.1" minor="0.1"/>
    </LV>
    ...
  </inputs>
  <outputs>
    <LV name="link">
      <LT name="low" type="Singleton" xm="0.0"/>
      <LT name="medium" type="Singleton" xm="0.5"/>
      <LT name="high" type="Singleton" xm="1.0"/>
      <rendering left="-0.1" right="1.1" major="0.2" minor="0.1"/>
    </LV>
  </outputs>
  <inference aggregation="MinMax" accumulation="Max" activation="Min">
    <rule name="R1">IF verb IS high AND position IS positiveSmall AND root IS true THEN link IS high.</rule>
    <rule name="R2">IF verb IS low AND root IS true THEN link IS low.</rule>
    <rule name="R3">IF noun IS high AND relativePosition IS negativeSmall THEN link IS high.</rule>
    <rule name="R4">IF noun IS high AND relativePosition IS positiveSmall THEN link IS high.</rule>
    <rule name="R5">IF det IS high AND nounParent IS high AND relativePosition IS negativeSmall THEN link IS high.</rule>
    <rule name="R6">IF det IS high AND nounParent IS high AND relativePosition IS positive THEN link IS low.</rule>
    <rule name="R7">IF det IS high AND nounParent IS low THEN link IS low.</rule>
  </inference>
  <defuzzyfication method="Centroid"/>
</fuzzy>
```

# Appendix C

# Class Diagrams

## C.1 Control

Figure C.1


CorpusManagement
- Corpus corpus
- LinkedList<String> lines
- int currentLineIndex
- int currentPhraseNumber
- int chapterId
+ Corpus getCorpus()
+ int getLevel(String)
+ void readArticle(Article)
+ void refreshCorpus()
- String convertLineToTitle(String)
- String postprocessPhrase(String, String)
- boolean importArticleFromXML(Article, File)
- void exportArticle2XML(Article, File)
- void readChaptersRecursively(Chapter)
TextPreprocessingControl
- PreprocessedPhrase preprocessPhrase(String)
- boolean importDependencyGraphFromXMLRecursively(Node)
OWLConnector
- boolean writeOwl(Ontology, String)
InformationExtractionControl
Ontology ontology
- final ConceptExtraction conceptExtraction
- final HierarchyOfConceptsExtraction hierarchyOfConceptsExtraction
- final RelationExtraction relationExtraction
+ Ontology getOntology()
+ void extractConcepts(Corpus, double)
+ void setOntology(Ontology)
+ void synthesizeOntology()
HierarchyOfConceptsExtraction
relationExtraction
ConceptExtraction
- LinkedList<Concept> conceptsCandidates
- LinkedList<Concept> conceptsEstimation
- LinkedList<Relation> synonymConcepts
- long localNumberOfWords
- LinkedList<Concept> calculateConceptsEstimation(double)
- LinkedList<Concept> getConceptCandidates()
- LinkedList<Concept> getConceptsEstimation()
- long getLocalNumberOfWordsRecursively(Chapter)
- long getLocalWordFrequency(Corpus, String)
- void getConceptCandidatesRecursively(Corpus, Chapter)

## C.2 Preprocessing

Figure C.2

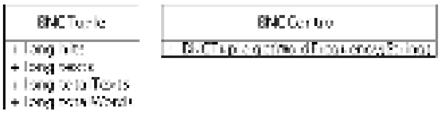


Figure C.3

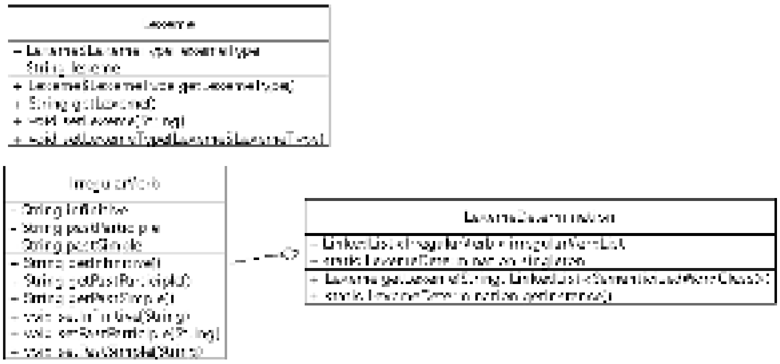

Lexeme
- Lexeme$LexemeType lexemeType
- String lexeme
+ Lexeme$LexemeType getLexemeType()
+ String getLexeme()
+ void setLexeme(String)
+ void setLexemeType(Lexeme$LexemeType)
IrregularVerb
- String infinitive
- String pastParticiple
- String pastSimple
+ String getInfinitive()
+ String getPastParticiple()
+ String getPastSimple()
+ void setInfinitive(String)
+ void setPastParticiple(String)
+ void setPastSimple(String)
LexemeDetermination
- LinkedList<IrregularVerb> irregularVerbList
- static LexemeDetermination singleton
+ Lexeme getLexeme(String, LinkedList<SemanticLexWordClass>)
+ static LexemeDetermination getInstance()

Figure C.4

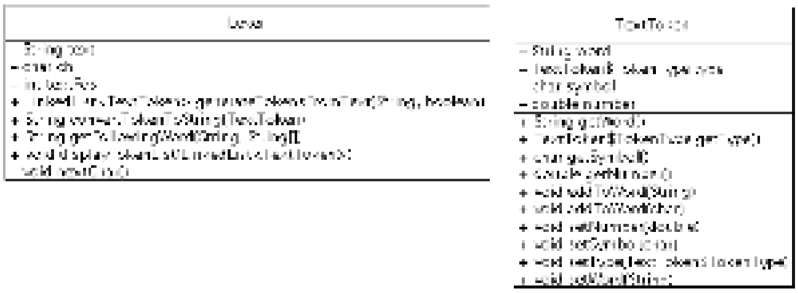

Lexer
- String text
- char ch
- int textPos
+ LinkedList<TextToken> generateTokensFromText(String, boolean)
+ String convertTokenToString(TextToken)
+ String getFollowingWord(String, String[])
+ void displayTokenList(LinkedList<TextToken>)
- void nextChar()
TextToken
- String word
- TextToken$TokenType type
- char symbol
- double number
+ String getWord()
+ TextToken$TokenType getType()
+ char getSymbol()
+ double getNumber()
+ void addToWord(String)
+ void addToWord(char)
+ void setNumber(double)
+ void setSymbol(char)
+ void setType(TextToken$TokenType)
+ void setWord(String)

Figure C.5

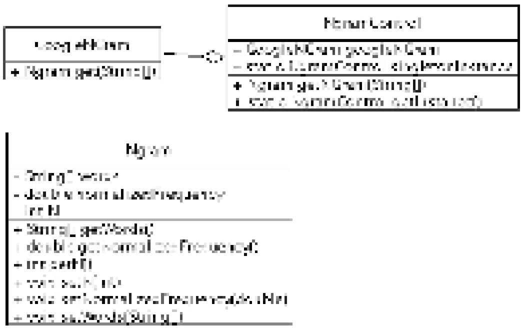

GoogleNGram
+ Ngram get(String[])
NgramControl
- GoogleNGram googleNGram
- static NgramControl singletonInstance
+ Ngram getNGram(String[])
+ static NgramControl getInstance()
Ngram
- String[] words
- double normalizedFrequency
- int N
+ String[] getWords()
+ double getNormalizedFrequency()
+ int getN()
+ void setN(int)
+ void setNormalizedFrequency(double)
+ void setWords(String[])

Figure C.6

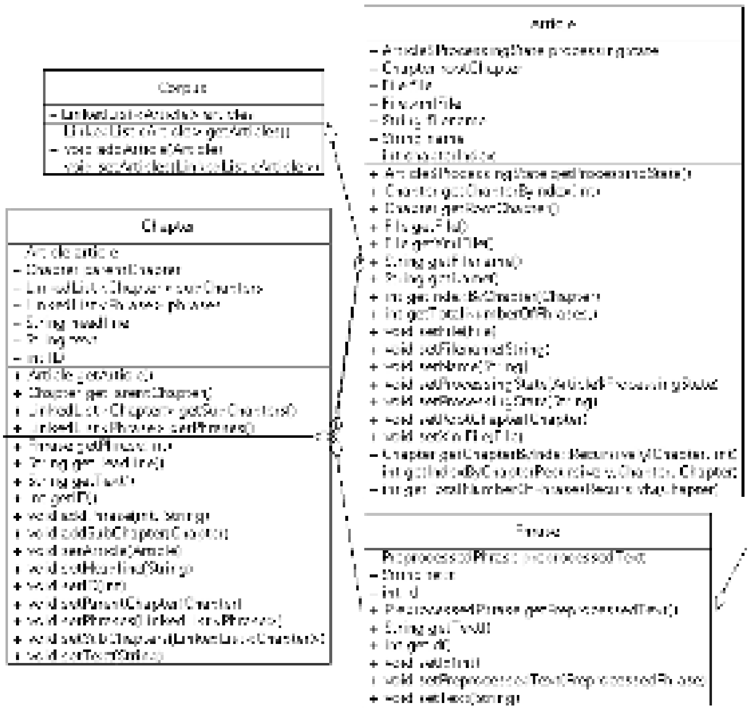


Figure C.7

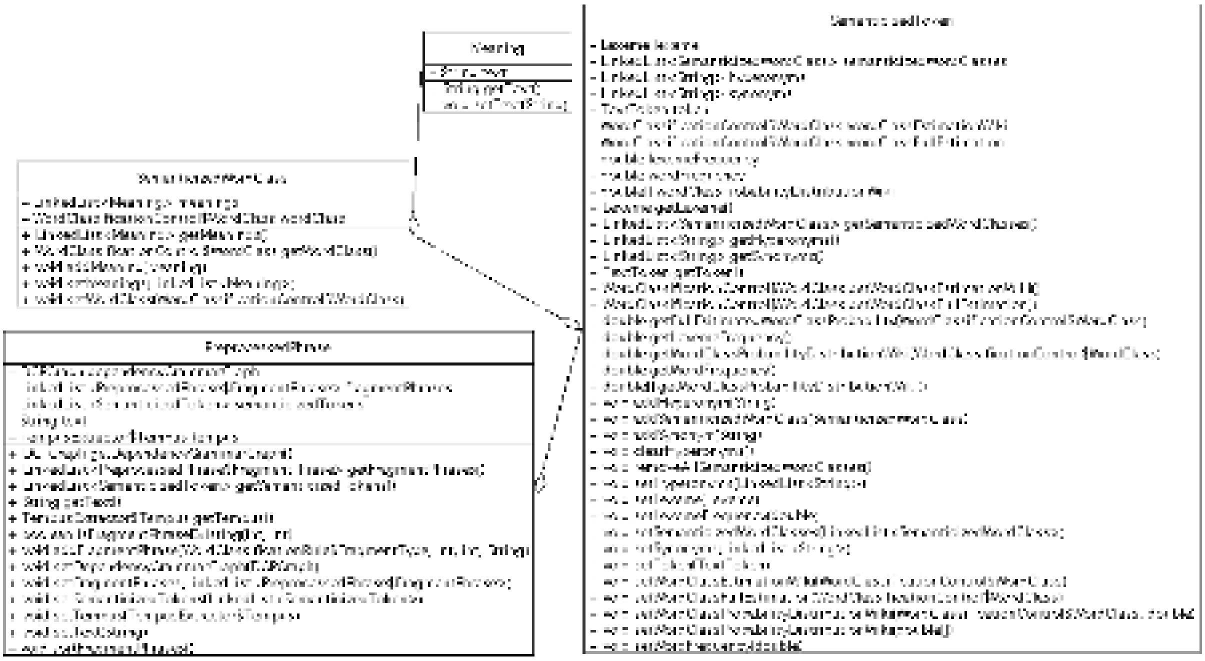


Figure C.8

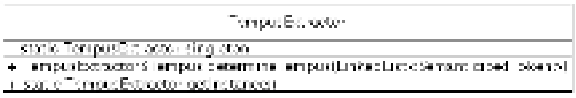

TempusExtractor
- static TempusExtractor singleton
+ static TempusExtractor getInstance()

Figure C.9

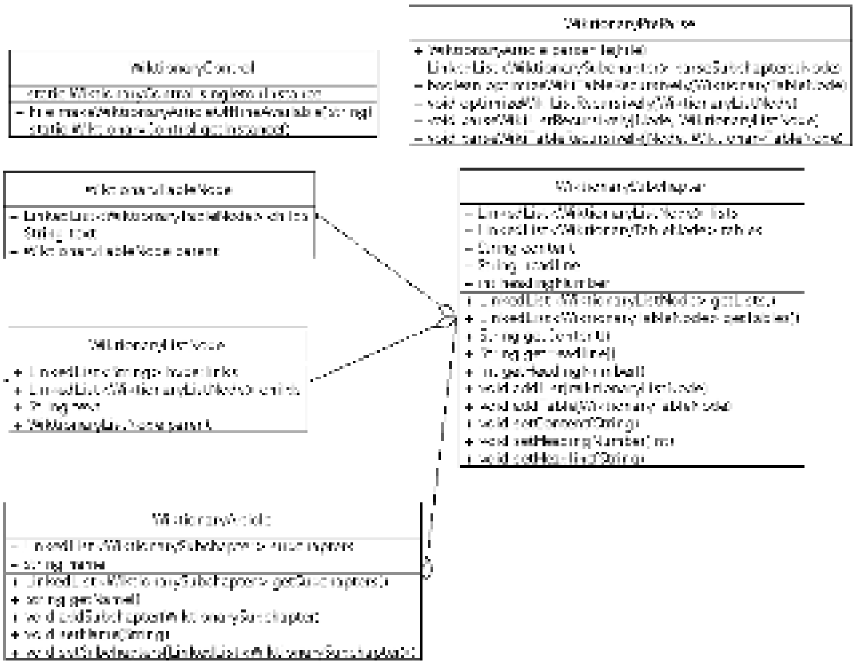

WiktionaryControl
- static WiktionaryControl singletonInstance
+ File makeWiktionaryArticleOfflineAvailable(String)
+ static WiktionaryControl getInstance()
WiktionaryParser
+ WiktionaryArticle parse(String)
- LinkedList<WiktionarySubchapter> parseSubchapters(Node)
- boolean optimizeWikiTableRecursively(WiktionaryTableNode)
- void optimizeWikiListRecursively(WiktionaryListNode)
- void parseWikiListRecursively(Node, WiktionaryListNode)
- void parseWikiTableRecursively(Node, WiktionaryTableNode)
WiktionaryTableNode
- LinkedList<WiktionaryTableNode> childs
String text
- WiktionaryTableNode parent
WiktionarySubchapter
- LinkedList<WiktionaryListNode> lists
- LinkedList<WiktionaryTableNode> tables
- String content
- String headline
- int headingNumber
+ LinkedList<WiktionaryListNode> getLists()
+ LinkedList<WiktionaryTableNode> getTables()
+ String getContent()
+ String getHeadline()
+ int getHeadingNumber()
+ void addList(WiktionaryListNode)
+ void addTable(WiktionaryTableNode)
+ void setContent(String)
+ void setHeadingNumber(int)
+ void setHeadline(String)
WiktionaryListNode
+ LinkedList<String> hyperlinks
+ LinkedList<WiktionaryListNode> childs
+ String text
+ WiktionaryListNode parent
WiktionaryArticle
- LinkedList<WiktionarySubchapter> subchapters
- String name
+ LinkedList<WiktionarySubchapter> getSubchapters()
+ String getName()
+ void addSubchapter(WiktionarySubchapter)
+ void setName(String)
+ void setSubchapters(LinkedList<WiktionarySubchapter>)

Figure C.10

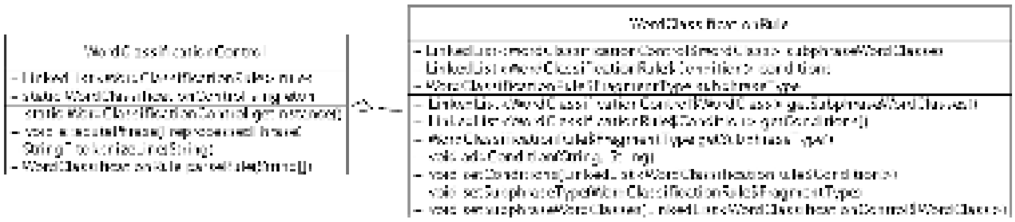

WordClassificationControl
- LinkedList<WordClassificationRule> rules
- static WordClassificationControl singleton
static WordClassificationControl getInstance()
- void annotatePhrase(PreprocessedPhrase)
String[] tokenize(String)
- WordClassificationRule parseRule(String[])
WordClassificationRule
- LinkedList<WordClassificationControl$WordClass> subphraseWordClasses
LinkedList<WordClassificationRule$Condition> conditions
- WordClassificationRule$FragmentType subphraseType
- LinkedList<WordClassificationControl$WordClass> getSubphraseWordClasses()
- LinkedList<WordClassificationRule$Condition> getConditions()
- WordClassificationRule$FragmentType getSubphraseType()
void addCondition(String, String)
- void setConditions(LinkedList<WordClassificationRule$Condition>)
void setSubphraseType(WordClassificationRule$FragmentType)
- void setSubphraseWordClasses(LinkedList<WordClassificationControl$WordClass>)

Figure C.11

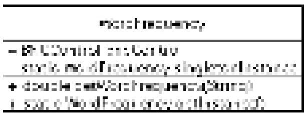

WordFrequency
- DBControl dbControl
static WordFrequency singletonInstance
+ double getWordFrequency(String)
+ static WordFrequency getInstance()

Figure C.12

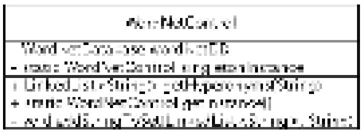
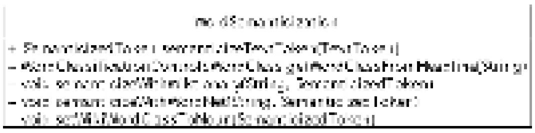
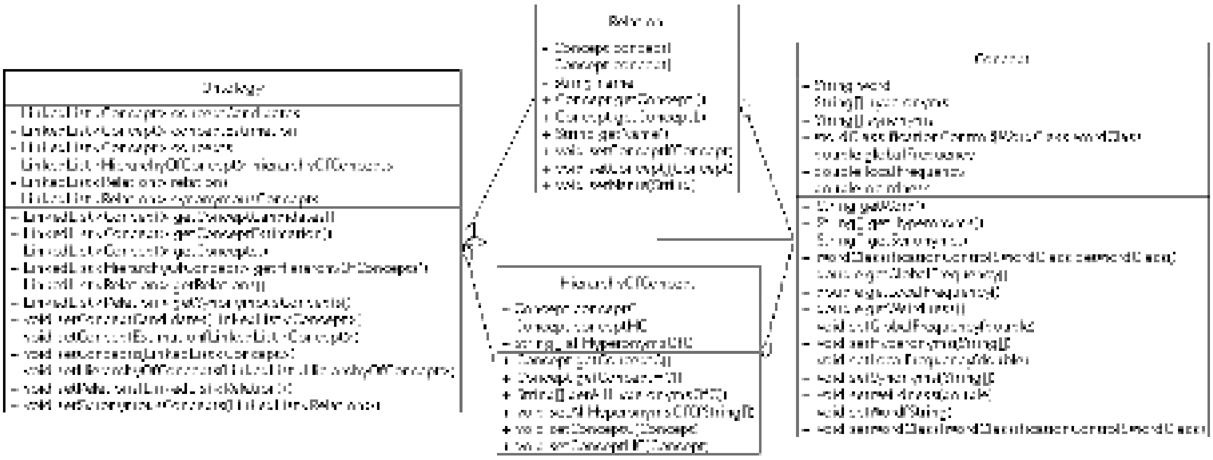

WordNetControl
WordNetDatabase wordNetDB
- static WordNetControl singletonInstance
+ LinkedList<String> getHypernyms(String)
+ static WordNetControl getInstance()
- void addStringToSet(LinkedList<String>, String)

Figure C.13


WordSemanticization
+ SemanticizedToken semanticizeTextToken(TextToken)
- WordClassificationControl$WordClass getWordClassFromHeadline(String)
- void semanticizeWithWiktionary(String, SemanticizedToken)
- void semanticizeWithWordNet(String, SemanticizedToken)
void setWithWordClassToNoun(SemanticizedToken)

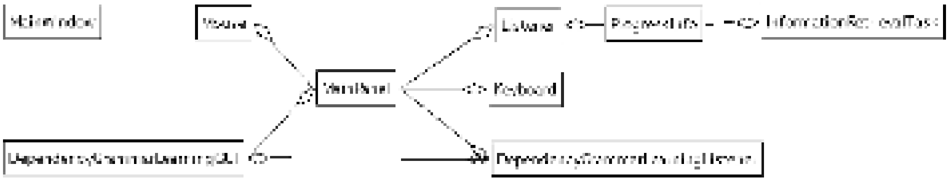
## C.3 Information Extraction

Figure C.14



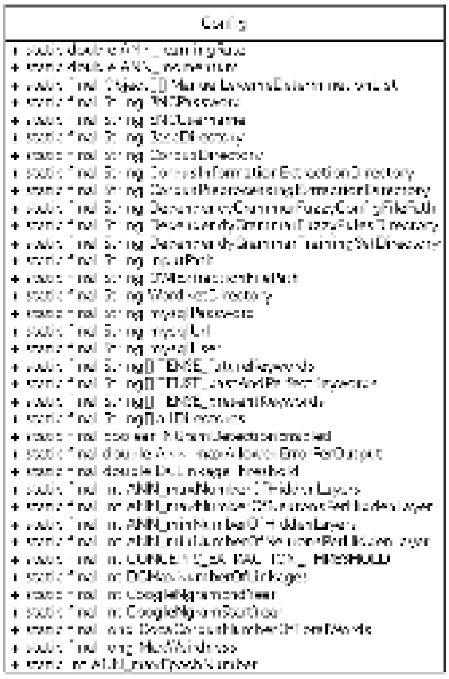
## C.4 Graphical User Interface

Figure C.15



## C.5 Other Classes

Figure C.16



Config
+ static double ANN_learningRate
+ static double ANN_momentum
+ static final String DBCPassword
+ static final String DBCUsername
+ static final String BaseDirectory
+ static final String CorpusDirectory
+ static final String CorpusInformationExtractionDirectory
+ static final String CorpusPreprocessingExtractionDirectory
+ static final String InputPath
+ static final String WordNetDirectory
+ static final String mysqlPassword
+ static final String mysqlUrl
+ static final String mysqlUser
+ static final String[] TENSE_futureKeywords
+ static final String[] TENSE_presentKeywords
+ static final boolean NGramDetectionEnabled
+ static final int ANN_maxNumberOfHiddenLayers
+ static final int ANN_minNumberOfHiddenLayers
+ static final int CONCEPT_EXTRACTION_THRESHOLD
+ static final int GoogleNgramStartYear
+ static final int GoogleNgramEndYear
+ static int ANN_maxEpochNumber

Figure C.17

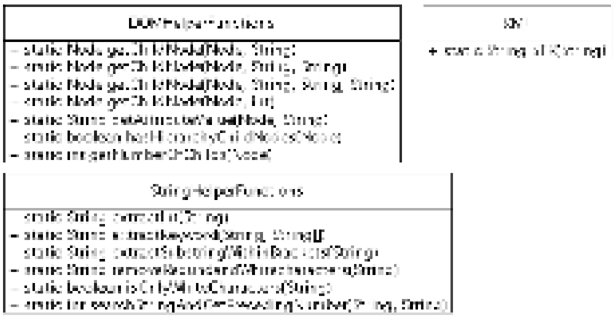

DOMHelperFunctions
- static Node getChildNode(Node, String)
- static Node getChildNode(Node, String, String)
- static Node getChildNode(Node, String, String, String)
- static Node getChildNode(Node, int)
- static String getAttributeValue(Node, String)
static boolean hasHierarchyChildNodes(Node)
- static int getNumberOfChildren(Node)
XML
StringHelperFunctions
static String extractKeyword(String, String[])
static String extractSubstringWithinBrackets(String)
- static String removeRedundantWhitecharacters(String)
- static boolean isOnlyWhiteCharacters(String)
- static int searchStringAndGetPrecedingNumber(String, String)

Figure C.18

Tests
Main
+ static void main(String[])

# List of Figures

# List of Tables