

Univerzita Karlova v Praze

Filozofická fakulta

Ústav informačních studií a knihovnictví

Informační věda

Roman Chýla

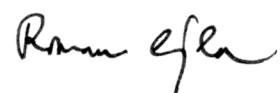
**Automated methods of textual content analysis and
description of text structures**

Automatizované metody popisu struktury odborného textu a vztah některých prvků ke
kvalitě textu

Rok podání: 2011

Vedoucí práce: Doc. PhDr. Vladimír Smetáček, CSc

Prohlašuji, že jsem dizertační práci napsal samostatně a s využitím pouze uvedených a řádně citovaných pramenů a literatury a že práce nebyla využita v rámci jiného vysokoškolského studia či k získání jiného nebo stejného titulu.

A handwritten signature in black ink, appearing to read "Roman Efla". The script is cursive and fluid.

Ženeva, 9. srpna 2011

Abstract

Universal Semantic Language (USL) is a semi-formalized approach for the description of knowledge (a knowledge representation tool). The idea of USL was introduced by Vladimir Smetacek in the system called SEMAN which was used for keyword extraction tasks in the former Information centre of the Czechoslovak Republic. However due to the dissolution of the centre in early 90's, the system has been lost.

This thesis reintroduces the idea of USL in a new context of quantitative content analysis. First we introduce the historical background and the problems of semantics and knowledge representation, semes, semantic fields, semantic primes and universals. The basic methodology of content analysis studies is illustrated on the example of three content analysis tools and we describe the architecture of a new system. The application was built specifically for USL discovery but it can work also in the context of classical content analysis. It contains Natural Language Processing (NLP) components and employs the algorithm for collocation discovery adapted for the case of cooccurrences search between semantic annotations.

The software is evaluated by comparing its pattern matching mechanism against another existing and established extractor. The semantic translation mechanism is evaluated in the task of automated document classification with special attention to the problem of semantic ambiguity and correct translation. Finally we evaluate the ability of the system to discover statistically significant semantic relationships from textual corpora.

Abstrakt

Univerzální sémantický jazyk (USJ) je semi-formalizovaný způsob zápisu znalostí (systém pro reprezentaci znalostí). Myšlenka USJ byla rozvinuta Vladimírem Smetáčkem v 80. letech při pracích na systému SĚMAN (Univerzální sémantický analyzátor). Tento systém byl využíván pro automatizovanou extrakci klíčových slov v tehdejší informačním centru ČSSR. Avšak se zánikem centra v 90. letech byl systém SEMAN ztracen.

Tato dizertace oživuje myšlenku USJ v novém kontextu automatizované obsahové analýzy. Nejdříve prezentujeme historický kontext a problémy spojené s reprezentací znalostí, sémů, sémantických polí, sémantických primitivů a univerzálií. Dále je představena metodika kvantitativní obsahové analýzy na příkladu tří klasických aplikací. Podrobně popíšeme architekturu nové aplikace, která byla vyvinuta speciálně pro potřeby evaluace USJ. Program může fungovat jako nástroj pro klasickou obsahovou analýzu, avšak obsahuje i nástroje pro zpracování přirozeného jazyka (NLP) a využívá algoritmy pro vyhledávání kolokací. Tyto byly upraveny pro potřeby vyhledávání vazeb mezi sémantickými anotacemi.

Jednotlivé součásti programu jsou podrobeny praktickým testům. Subsystem pro vyhledávání vzorů v textech je porovnán s existujícím extraktorem klíčových slov. Mechanismus pro překlad do sémantických kódů je otestován na příkladu automatické klasifikace dokumentů a speciální pozornost věnujeme problémům mnohoznačného překladu. Poslední část evaluace se zabývá schopnostmi systému objevit významné vazby mezi sémantickými anotacemi v textu.

Acknowledgements

First of all, I want to thank my supervisor Vladimir Smetacek, who helped me to see many things differently. He patiently bore all my doubts and valantly fought off all attacks in the long discussions we had with a mischievous smile, saying; “Maybe it is all useless and we'll have to junk it in the end. But who knows for sure before trying?” For this and for much more I owe him sincere gratitude.

A great thanks goes to friends who helped me with questions, helped me review the text, who did the proofreading or simply listened and discussed with me various problems. In no particular order they were Claire Palmiste, Hugo Day, Linda Skolkova, Semiray Girgis, Ilknur Hos, Piotr Praczyk, Kvetoslav Minarik, Ariane Koek. Thank you. Thank you!

And of course to my parents! How could I ever finish this work without my mom's gentle pressure and my father's kind benevolence? If I was able to complete the task, it is thanks to what they thought me and what they helped to shape in me. This thesis is a product of a long time which was made longer by my stay in different countries. But I dare say it was also a lot more interesting that way. I learnt a lot. And in the end it will be all put to a good use.

Table of Contents

I. INTRODUCTION.....	1
I.1 Motivation	2
I.2 Research questions.....	3
I.3 Contributions.....	4
I.4 Thesis outline.....	4
II. SEMANTICS AND KNOWLEDGE REPRESENTATION.....	7
II.1 USL and Formal Concept Analysis. .10	
II.2 Existential-conjunctive logic.....	12
II.3 Semantics.....	14
II.3.1 What is meaning?.....	14
II.3.2 Semantic fields.....	16
II.3.3 Semantic primes and universals.....	19
II.4 USL and Lexical semantics.....	23
II.4.1 Words as lexical items.....	24
II.4.2 But what is a word? And how do we define its meaning?.....	25
II.4.3 Words and grammatical categories.....	26
II.4.4 Problematic areas of the thesaurus.....	27
II.4.4.1 Homonymy.....	27
II.4.4.2 Polysemy.....	27
II.4.4.3 Synonymy.....	28
II.4.4.4 Antonymy.....	28
II.4.4.5 Hyponymy/hypernymy.....	29
II.4.4.6 Meronymy.....	29
II.4.4.7 Difference between names and types.....	29
II.4.5 Word sense disambiguation.....	30
II.5 Concluding remarks on USL.....	31
III. CONTENT ANALYSIS.....	35
III.1 Main types of content analysis.....	39
III.2 The components of content analysis... 40	
III.2.1 Coding and categories.....	41
III.2.1.1 Reliability and validity.....	43
Validity.....	45
III.2.2 Data	46
Sampling units.....	47
Recording units.....	47
Units of analysis.....	47
Contextual units.....	47
IV. SOFTWARE FOR CONTENT ANALYSIS. 49	
IV.1 General inquirer.....	53
IV.1.1 Disambiguation rules.....	55
IV.1.2 Dictionary.....	56
IV.2 TABARI.....	60
IV.2.1 What TABARI does.....	61
IV.2.2 How TABARI works.....	62
1. Word classification.....	65
2. Processing local grammatical structures.....	65
3. Event coding.....	65
4. Information output.....	66
IV.2.3 Pattern matching.....	68
IV.2.4 Dictionary.....	71
IV.2.5 Concluding remarks.....	72
IV.3 Yoshikoder.....	73
IV.3.1 What Yoshikoder does.....	74
IV.3.2 How Yoshikoder works.....	76
IV.3.3 Concluding remarks.....	78
IV.4 SEMAN.....	78
IV.4.1 NLP pre-processing.....	79
IV.4.2 Translation into semantic codes.....	83
IV.4.3 Storage and analysis.....	89
IV.4.4 Search for statistically significant cooccurrences.....	92
IV.4.4.1 Types of cooccurrences.....	94
IV.4.4.2 Extracting cooccurrences from text	96
IV.4.4.3 Association measures.....	102
IV.4.5 Dictionary creation and maintenance.....	106
IV.4.6 SEMAN GUI and scripting control. .	109
IV.5 Comparison.....	112
V. EVALUATION OF SEMAN.....	115
V.1 The pattern matching mechanism...115	
V.1.1 BibClassify.....	115
V.1.2 Comparison.....	119
V.2 Semantic ambiguity.....	124
V.2.1 Feature selection and scaling.....	127
V.2.2 The comparison.....	129
V.2.2.1 HEP corpus	134
V.2.2.2 The 20 newsgroups corpus.....	137
V.2.3 Search for significant combinations. .	142
V.2.3.1 Results.....	144
V.2.3.2 Discussion.....	147
VI. CONCLUSIONS.....	151
VII. BIBLIOGRAPHY.....	155
VIII. APPENDICES.....	162
VIII.1 Illustration Index.....	162
VIII.2 Bibliography on SEMAN.....	163
VIII.3 Installation instructions.....	165

I. INTRODUCTION

“Vague, but exciting.” Such was the handwritten note of a reviewer on a proposal by Tim Berners-Lee for a system that later became known as World Wide Web; the biggest network people have ever built and also the biggest source of information for and about people. It was designed for researchers at European Organization for Nuclear Research (CERN), but exceedingly elegant and beautiful in its simplicity, it was adopted later by people outside the research community. They built on top of it and created more than just a sum of the individual parts. And as with everything in human history, this tool was put to use for all possible zeals – it virtually saves lives, and literally takes them away as well.

But I do not mention the beginnings of WWW to draw any sort of a far fetched and arrogant parallel between the big invention called Web and the small research task that I am going to present here. No, not at all. I have other reasons. I owe much to the Web and to “IT” at CERN. A strange twist of luck or chance brought me to CERN and these lines are written only a few meters away from the shabby and dark corridor where Tim Berners-Lee and Robert Cailliau once sat in front of the first web browser. It is a dark passage down there, but visitors must notice the golden plaque on the wall – the only source of light in the dark alley. When I came closer to it another handwritten note came to my mind. It was attached to an old computer in the CERN museum and read; “This machine is a server. DO NOT POWER DOWN!!!”¹

As you can imagine, the beginnings must have been difficult and not always smooth. The idea of the global information system must have been exciting, perhaps the more vague, the more exciting. But the initial rush was probably soon replaced by a realization that things must be implemented. Not “somehow” or “anyhow”, but well enough. To do more than necessary is as wrong as to do less than necessary. And there it is where the parallel story for me starts. It was in classes and during discussions with my supervisor at Charles University. Discussions about a computer system that could “do something useful” - something that could extract, filter out, provide useful information, make sense of the information avalanche in which we are submerged. It was during these discussions about history, religion, all the biological, ethical, rational or irrational decisions that people make, that the idea of a computer system called SEMAN was introduced to me. And the more vague, the more exciting it was!

What if we were able to “read” the texts written by people and indirectly deduce from these texts what and how people think? What if we use the information now widely available on the Web to analyse ourselves? What if we could help the computer to make some sense of the bits that are floating around? Dreams, for sure, vague but exciting dreams, I dare say.

¹ http://en.wikipedia.org/wiki/File:First_Web_Server.jpg

My supervisor started a work on such a project nearly 35 years ago, in the times when mainframes (and especially mainframe computers in the then communist block) had very limited memory. Software and hardware constraints were considerable. Yet it was still possible to implement the idea of a computer code which translated the natural language texts into a semi-formalized form that got analyzed later. Vladimir Smetacek called this form “Universal Semantic Language” (USL) and it allowed him and his collaborators to look at the features of texts. USL was effectively a knowledge representation system which encoded our knowledge of the world into formalized categories (that stood for concepts). Words were used as pointers towards concepts. In the semantic triangle we had the word on one edge, semantic codes on the other edge, and indirect link of a meaning between the two.

The system used to work in the centre of information industry of the Czechoslovak Republic. It extracted keywords from the abstracts of scientific papers, but with the dissolution of the centre in the post-communist times, the information system as well all the products of the centre were lost. A few attempts were made to revive the idea, but as far as I know, all were unsuccessful which brings us to the beginning of this thesis.

I.1 MOTIVATION

The main motivation of this research was to build a system that would implement the USL and will test its capabilities. It also wanted to verify assumptions we had had about its ability to extract interesting data.

A concrete application of such an idea is perhaps not useless. In offices and reading rooms, thousands of researchers and analysts every day grapple with the textual data and ask questions about them. We have at our disposal extraordinary amounts of text, but our ability to process them is limited. Perhaps if we were able to find what people wrote about the reality, then one day we could deduce from these facts what was indeed happening in reality. For example, if there was a high degree of violence in society, then the products of society (such as texts) could testify about this higher degree of violence.

Ideally, the computer programs should be able to “read and understand” these texts in the same ways as people do. But we know quite well this is just a dream and what seems so easy for people is still too difficult for any intelligent computer system. The reality is much more profane and really not that exciting as science fiction or sensational newspapers purport. There has been huge progress in many fields of artificial intelligence, robotics, language processing and computer science in general. Nevertheless we are still far away from the expectations that our ancestors had from computers just only 40, 30 or even 20 years ago (Shapiro, Tackett, Dawson, and Markoff 1998; Cuilenberg, Kleinnijenhuis, and de Ridder 1998; Krippendorf 2004a) But the progress came slowly in small, incremental changes and contributions such as this thesis might be perhaps useful as well.

I.2 RESEARCH QUESTIONS

I was facing the task of reviving the idea of a system that used to work in a different context. There was a body of published literature about the system, and the idea seemed appealing – but it was still a vague idea. The published literature was not new, the software did not exist anymore. But Dr. Smetacek hopefully remembered every detail. Nevertheless I had to recreate SEMAN myself as I understood it. We spent long afternoons in discussions and disputes whether something is a good idea or not, but usually the final answer to my doubts was: “we shall test it and see”.

So, there were many doubts that I had at the beginning of this work, today, arguably I know much more, but it is good to remember them.

Firstly, there were questions about the USL itself. As a tool for knowledge organization – where can the idea be traced from? Are we using something similar as people were/will be using after us? And there was this 'unpleasant' feeling of anarchy. The way the USL is described is somewhat 'messy' and it does not require nor impose strict rules. By design it refuses to have organizational principles (ideology). This was very disconcerting at the beginning. I was expecting some organization principle, certain philosophy and patterns of construction of the language. Because rule(s) are needed...? (Or are they?)

But the most important questions were about the application. Is it possible to translate the text written in the natural language into this flat structure of USL? Can we enhance the text with semantic tags around words? Can we do it despite all the problems of semantic ambiguity associated with the translation? Can we extract useful information from such processed texts? Will it all work?

In a more orderly fashion, the questions and tasks were as following:

1. Is the idea of the universal semantic language (USL) translatable to the form of a working application in a different domain?
2. Can we translate the texts from their natural form into the language of USL and actually avoid the problems of increasing entropy?
3. Can the process of extraction of facts work?
4. Is the current form of knowledge representation management sustainable? And what tools and methods one needs to develop to improve them?
5. Is it possible to have one, universal definition of a meaning?

These were the main questions for which the work on a new application with the old name (SEMAN) was started.

I.3 CONTRIBUTIONS

The thesis makes the following research contributions:

1. We have developed a new research tool
2. We have reviewed the theory behind the USL and summarized its application into the context of content analysis
3. We have evaluated the ability of the system to code texts and retrieve semantic relationships

Besides working on the application of USL, SEMAN was conceived also as a generic tool for content analysis. It is comparable in functionality to some existing tools and this is another contribution of the thesis.

I.4 THESIS OUTLINE

Chapter II. starts with the introduction to the problem of knowledge representation. We continue with the history of semantic primes and universals including the examples from philosophy and recent lexicography. The idea of USL is compared with the theory of Universal Characteristic, semantic fields, semantic primes and universals, and we describe the features of the USL representation discussing the data structures used for representation. The whole chapter II.3 is dedicated to a discussion on semantics and whether we could find one unique meaning/definition of concepts. The chapter concludes with personal remarks based on the experience with the maintenance of the USL dictionary. Remarks about its potential and shortcomings I was able to perceive.

The following chapters III. and IV. concentrate on content analysis. First we review the basic methodology of content analysis and the problems of “definition of meaning”. This chapter is followed by review of tools for content analysis; representative tools for so called “classical content analysis studies”. Right after this section comes the detailed description of SEMAN, the architecture of the software, components, storage mechanism and algorithms for translation and analysis. SEMAN is compared against the previous tools at the end of the chapter.

The last chapter V. represents the most important part of the thesis. We evaluate the performance of SEMAN. Firstly, the internal matching mechanisms which is a crucial component for the extraction of information. Next we conduct an experiment with document classification. The goal of this procedure is to find out whether SEMAN can deal with the semantic ambiguity in the process of translation from the natural language form into USL semantic codes. The final part of the evaluation consists of two experiments. We

want to find out whether SEMAN can recognize and extract significant cooccurences of semantic codes from the translated corpus and what proportion of significant pairs may be missed.

II. SEMANTICS AND KNOWLEDGE REPRESENTATION

This chapter represents a short discourse on history of semantic fields and semantic primitives, the main building blocks of USL. We will also discuss various problems associated with the representation of knowledge, touching upon technical aspects of the primitives and their representation in computer memory in later chapters. But before we delve into semantics, we can look at the world of ideas and their representations in the philosophy. We will start with Aristotle and his endeavour to organize the knowledge of a physical world. Aristotle was in many fields the first, he compiled an unprecedented body of facts. And what is more, Aristotle devised a system of their categorisation. Being opposed to the world of *ideas* of his teacher Plato, he saw the origin of things in the physical world and the *idee* as a reflection of it; he connected the physical world with the static, spiritual heaven and made of it a dynamic, organized encyclopaedia of facts.

Aristotle was the first to establish terminology for logic, physics, biology, linguistics and many other fields. Out of the invented terms, the *categories*, *quantity*, *quality*, *genus*, *species*, *noun*, *verb*, *subject*, *predicate* are the most important for the subject of our study. Also Aristotle's fundamentals of syllogisms and deductive logic are of no lesser importance, such as:

```
All humans are mortal.
Socrates is human.
Therefore Socrates is mortal.
```

Much later, the four main syllogistic patterns together with the fifteen derived patterns were studied by Christian scholastics in the many centuries that followed the dissolution of the Western Roman empire. They served as a basis for logical operations and evolved into rich set of tools and theories ranging from the binary, two-typed predicate logic into fuzzy and higher-order logics, mathematical logic of the 20th century and also in the formal semantics of the Montague grammar. We will dedicate more time to this topic when discussing properties of the *universal semantic language* but let us first address the problem of *knowledge representation*.

The first graphical notation of knowledge appeared very early, in the third century A.D., as a commentary on Aristotle's Categories by the philosopher Porphyry and is up to now called *Tree of Porphyry*.

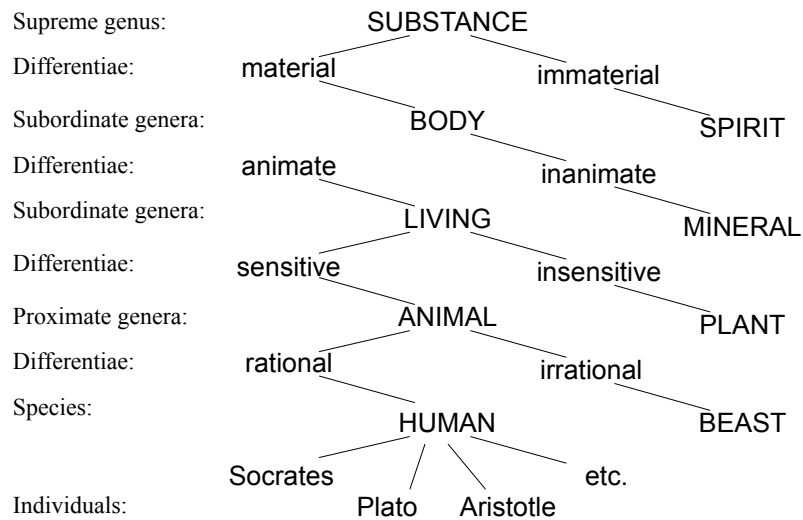


Illustration 1: Tree of Porphyry

Substance, for instance, is the single highest genus of substances, for no other genus can be found that is prior to substance. Human is a mere species, for after it come the individuals, the particular humans. The genera that come after substance, but before the mere species human, those that are found between substance and human, are species of the genera prior to them, but are genera of what comes after them. (Sowa 2000, 4)

Up to current times, the features that distinguish the different species of the same genus are used in the construction of dictionaries, for definition of new categories inside Artificial Intelligence (AI) systems, as well as for object-oriented design and programming. The tree shows the important property of *inheritance*, which in other words means there is a **path** from the current element towards its ancestors.

This is the principle of **composition** and **specialization** that we seem to find in nature around us, and that we, together with Aristotle, project onto the models of the world as we understand it.

While science flourished on the Arab peninsula and Indian continent, the European scholastics were living in the Dark Ages. In our short historical sketch, we will leave them there and jump fourteen centuries forward to the work of the German philosopher, diplomat and mathematician Georg Wilhelm Leibniz who, amongst many other things, developed *Universal Characteristic*. A tool for knowledge description and knowledge representation¹. Leibniz used combinatorics of prime numbers to define complex concepts. First he defined a few *primitive concepts*:

substance = 2	
material = 3	immaterial = 5
animate = 7	inanimate = 11
sensitive = 13	insensitive = 17
rational = 19	irrational = 23

¹ We shall not forget Leibniz was also a librarian.

Complex concepts were then represented by a product of *primitives*. In Leibniz's theory, for instance, “body” is material substance. The characteristics of the concept body are therefore represented by the product of 2 and 3, similarly the number for mineral is $2 \times 3 \times 11$, or 66; and the number for human is $2 \times 3 \times 7 \times 13 \times 19$, or 10.374.

The number represents all the composite characteristics of the concept and Leibniz defined operations for reasoning about them. To test whether human has body, the number 10.374 must be exactly divisible by 6. And if human is insensitive the number 10374 must be exactly divisible by 17, and it is not. Therefore no human is insensitive, at least for this definition of the concept.

Despite the development of mechanical calculators in the seventeenth century, no practical application of *Universal Characteristic* was possible due to large prime numbers.² It was only centuries later that a similar approach would not present insuperable problems. Nevertheless the principle was established. And it keeps returning in many knowledge representation systems, be they lattices, USL or decimal classification systems, for example:

Another representation, which is isomorphic to Leibniz's products of primes, represents each concept type by a string of bits. If n is the number of defining features, each product of primes is mapped to a string of bits of length n . A feature represented by the i -th prime number is mapped to a string with 1 in the i -th position and all other bits 0. Each concept type is represented by the logical OR of all the bit strings for each of its defining attributes. Concept type A is a subtype of B ($A < B$) if each position that has a 1 bit for b has a 1 bit for a . The lattice operators are defined as follows:

$A \cap B$ is the logical AND of the bit strings for a and b ;

$A \cup B$ is the logical OR of the bit strings for a and b . (Rajman 2007, 240)

Leibniz Combinatorics were also very important for the study of semantics and meaning, of which the detailed history is given in (Wierzbicka 1996). “Leibniz even began a program of lexical investigation with a view to discovering the primitive notions and rules of composition from which all complex notions were composed... his *ars combinatoria* or 'universal characteristic', is a direct ancestor of the present work [on semantic primes and universal].” (Goddard 1994, 9)

2 Interestingly, Leibniz is quoted as saying: "It is unworthy of excellent men to lose hours like slaves in the labour of calculation which could safely be relegated to anyone else if machines were used." [WIKI#History of computing hardware, quoting Smith, David Eugene (1929), *A Source Book in Mathematics*, New York: McGraw-Hill, pp. 180–18]

II.1 USL AND FORMAL CONCEPT ANALYSIS

The symmetric hierarchies generated by Leibniz's method are called *lattices* and they are an important method for knowledge organization in Artificial Intelligence. Another method with similar purpose called *Formal Concept Analysis* was invented by Rudolf Wille in 1984 (FCA).³ The difference is that Universal Characteristic was designed to describe in a compact way the properties of objects, while FCA approaches the same problem from the opposite direction. Its aim is to discover the underlying structure inside the set of known properties.

FCA is used for tasks such as automatic concept clustering and automated ontology construction. It is based on the Aristotelian duality of *intension* and *extension* – as in the classical logic, where intension defines the features (*differetiae*) that characterize objects, and extension is the set of all objects that belong to the given group.

For example, the concept type “Dog” applies to fewer entities in the real world than its supertype “Animal”, but more attributes are required to describe it. Concept are described as a natural way of grouping things by means of relationships between objects and attributes. The world (called *context*) is characterized by the three dimensions $\langle O, A, H \rangle$ where

- O is a set of objects;
- A is a set of features;
- H is the relationship between objects and attributes: $(o, a) \in H$ means that the object o has the attribute a..

This method is effective since it bypasses the combinatorial explosion induced by Leibniz's method. Leibniz's method generates lattices containing all possible combinations of features, but most of these combinations never actually occur in practice...The FCA lattices, however, contain only known concept types and likely generalizations...An FCA lattice can also be refined. For instance, new concepts can be created by adding supplementary features to the existing ones. Concept refinement corresponds to the application of the lattice operator $\cap$. For instance, “Beer” would be $\text{"sparkling"} \cap \text{"alcoholic"} \cap \text{"madeFromFrain"}$. (Rajman 2007, 240)

In this sense, USL can be viewed as informal application of FCA to lexical items, objects are represented by the entries in the lexicon and the semantic features are substituted by semantic codes. The relation H is denoted by the sign of equality and we can generate set of objects and their attributes.

³ Interestingly, the USL works with the same set of objects and attributes as FCA.

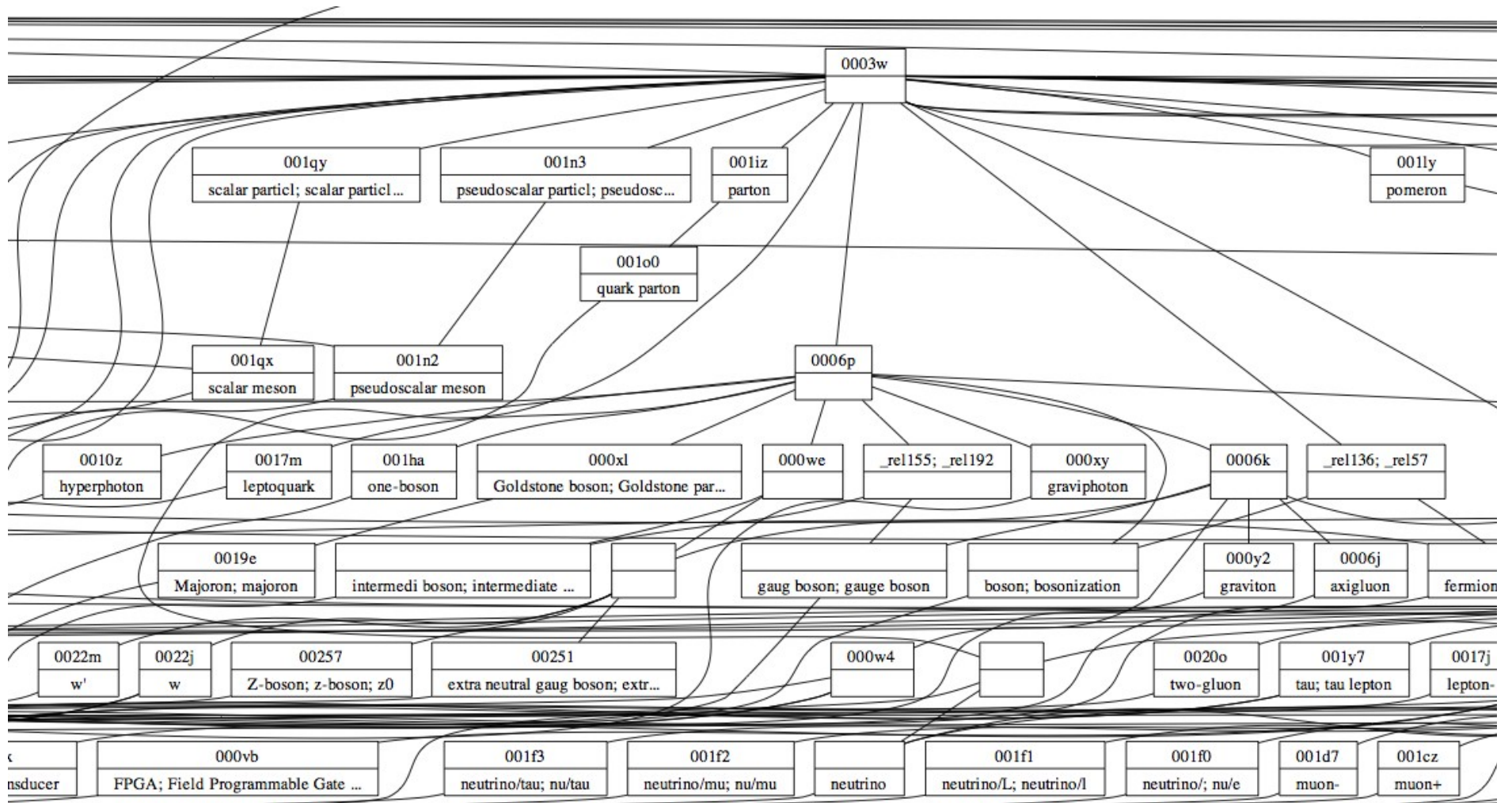


Illustration 2: Visualisation of the High-Energy physics taxonomy written in the USL. Some components are not attached to any word (like 0003w) and the FCA discovered two places where the entries miss connections (the empty boxes).

II.2 EXISTENTIAL-CONJUNCTIVE LOGIC

The subset of logic with only $\exists$ and $\wedge$ is called existential-conjunctive or EC logic. It is a common subset for translating, relating, and analyzing the specialized notations of many different fields. It is also the subset used to represent all the information stored in commercial database systems, both relational and object-oriented. EC logic is therefore an extremely important subset, but it has one serious limitation: it cannot represent any generalization, negations, implications, or alternatives. For that, the operators $\forall, \sim, \rightarrow$ and $\vee$ are necessary. (Sowa 2000, 17)

From the perspective of logic, USL can be classified as *existential-conjunctive logic*. As an example, the transliteration of the concept `father` in the USL into the notation of EC logic would be following:

$$(\exists x_1)(human(x_1) \wedge male(x_1) \wedge parenting(x_1))$$

This formula says that there exists an x that has the qualities denoted as *human*, *male* and *parenting*. If we were to include a more sophisticated version, we could specify arguments of a preposition.

$$(\exists x_1)(human(x_1) \wedge male(x_1) \wedge parenting(x_1, y_2))$$

This small subset of logic is sufficient to represent all the things that exist in the universe, their properties, and their relationships to every other thing. It forms the common logical core of every software system that stores and retrieves data of any kind. (Sowa 2000, 163)

USL is just a particular application of the EC logic and therefore shares the power of the EC logic. However, all those systems of knowledge representation have serious limitations. It was already mentioned when discussing the Leibniz's *Ars Combinatorics*. Multiplication can only represent conjunction, it cannot represent negation, disjunction, or implication. Even if Leibniz later replaced his single numbers with a pair of positive and negative numbers for each concept, he never found a way to represent all the logical operators and rules of inference.

Because of this limitation, we must draw a clear line between the power of a knowledge representation system that is based on the EC logic and the ontologies based on the higher order logics. Researchers in AI made this experience in early 80' when it became clear that knowledge representation systems needed to have two different parts – so called T-box and A-box (Schmolze, Beranek, and Inc 1985; Weida 1991). The *terminological reasoner* (T-box) has got task definitions of the terms (knowledge), while the *assertional reasoner* (A-

box) is responsible for drawing inferences, making assertions about those terms. Leibniz universals serve as a T-box system. It is a hierarchy of concepts with multiple inheritance. It holds characteristics of different various supertypes and has the power of EC logic. However, it is not the task of the T-box to encode every possible property of the world. It is the A-box, which uses this knowledge representation for automatic reasoning and inference-drawing operations. Either by employing specially crafted inference rules, logical constraints, machine learning methods or any other mechanism.

As we have previously mentioned, the EC logic can represent everything stored in a database, but it cannot represent negations, disjunctions, implications, or universal quantifiers. Thus USL cannot be charged with such functionality. Another external system must be developed. The application of the USL is limited and can only provide input for the second (A-box) system.

Predicate logic (or any other logic) is a simple language with only limited number of basic

symbols and operations, but the level of details depends on the choice of predicates, which at the end do not belong to the system itself. They represent an ontology of all relevant objects in the domain and finally they also have something to say about what kind of operations, what limitations and what constraints are applied. Thus some researchers will represent the knowledge from the perspective of logic and will try to define possible operations and conversions a priori. Others may see the problem from a different perspective. Rather than top-down, they will build the knowledge system from the bottom. This time it will not be achieved by theoretical conceptual analysis, but by a laborious process of concept-usage discovery. That is the case of bottom-up ontologies and also the case of SEMAN, as a particular application of the USL. With many iterations, the knowledge of the world that we have, is being slowly encoded and **clarified** inside the lexicon. This process is slow and laborious, and many difficulties are initially hidden. As Kant once mentioned: "If we were conscious of all that we know, we would have to be astonished at the great multitude of our cognitions." (Kant 1992, 569)

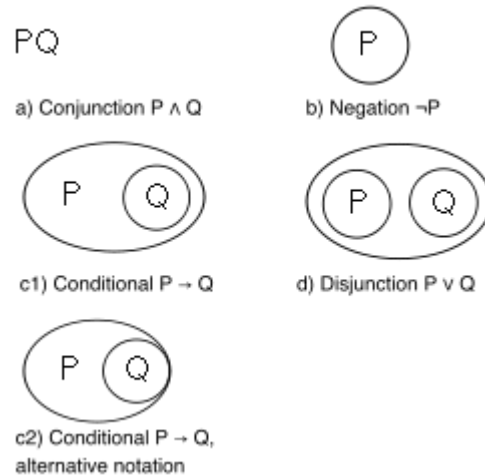


Illustration 3: A-box visually, using the input from the T-box, notation of existential graphs by C.S. Peirce

II.3 SEMANTICS

The crucial component of the search for components in the knowledge representation system is the question of meaning. What is meaning? While intuitively, this problem may seem trivial, it is so only for very careless observers. To anyone who has to deal with meaning, its definition, analysis, and representation, the answer to the question is crucial. Very soon it has too many facets to be grasped, and constraints to remember. In this section we will place the USL into the context of the current theories of meaning and discuss the most important problems that the system using USL faces. In later chapters, problems specific to lexical semantics will be discussed.

II.3.1 What is meaning?

Researchers elaborated whole theories to provide an answer to such a “simple” question. Differences were partly due to different goals of various disciplines (especially remarkable if we compare fields of content analysis and linguistics) but the fact remains that there exists no simple and universal answer to the question “what is meaning?”. To make discussion easier, we can limit observations to the field of *semiotics* and its subparts.

Semantics, the second of the branches of semiotics, is often described as a discipline concerned with study of common “in language present” meaning. Whilst *pragmatics* is seen as a discipline which studies meaning in use. How people use signs to convey a message, how they interpret the message. The line between two fields is often blurred, and in modern approaches to cognitive linguistics the division into *syntax*, *semantics* and *pragmatics* is often seen as something arbitrary or artificial. Meaning is not conceivable as something abstract, something that can exist without the subject, without the person who “understands” it. Yet for our purposes it might still be beneficial to distinguish semantics as an area of study of meaning which is abstracted away from users. And pragmatics, on the other hand, as study of meaning with the extra component of relation to speakers and hearers.

There exist two mainstream approaches to meaning in semantics, one **denotational** and the other **representational**. The denotational approach studies meaning as if it was something autonomous. Meaning there corresponds to the real-world objective reality and it can be found, objectively, in a text (for example). The denotational approaches are represented by the formal model theory of meaning and by formal logic. Most often by the Montague grammar.⁴ They represent and study meaning through the formal apparatus of logic.

Denotational approaches, if successful, have another advantage: they escape the problem of circularity...if we interpret English in terms of a metalanguage, another set of symbols, then we have just translated from one language to another. This second language then needs a semantics, and so on. As we shall see, formal semanticists do translate a natural language like English into a second, local

⁴ The Montague grammar treats natural languages as a form of a formalized language, with all consequences of the formal logic approach.

language, but this translation is only part of the semantic analysis. This logical language is then semantically grounded by tying it to real-world situations. The aim of a denotational approach is not just to convert between representations: it seeks to connect language to the world. (Saeed 2009, 307)

In the other of approaches, meaning is not studied through the real world and its in-word description. But it is seen as a kind of a mental construct.

...for semanticists [of this sort] semantic analysis involves discovering the conceptual structure which underlines language. For such linguists the search for meaning is the search for mental representations. Formal semanticists on the other hand come at meaning from an another angle: for them a primary function of language is that it allows us to talk about the world around us... From this perspective, understanding the meaning of an utterances is being able to match it with the situation it describes. Hence the search for meaning, from the denotational perspective, is the search for how the symbols of language relate to reality. (Saeed 2009, 305-306)

This remains a problem together with the fact that the representational approach cannot take advantage of the elaborate system of logical reasoning. First and foremost, there is a profound philosophical opposition between the two approaches. The denotational branch is closer to linguistic structuralism and denies all psychological features. It accepts to study meaning only through relationships between lexical items. Meaning is considered to be an autonomous system, something that can be isolated and described. The representational approach, on the other hand, is prevalent in cognitive semantics and does take into account the extra-linguistic features and processes inside the person. There is a strong psychological dimension. The representational theories strive to construct mental models of the conceptual world and test such models using language data. Cognitive semantics is thus a part of cognitive linguistics and is characterized by its encyclopedic nature. The dense net of relations that outnumber relations in language itself. It maintains that linguistic meaning cannot be isolated and studied as a world on its own.

Any content or semantic analysis using the USL will inevitably struggle with this aspect of knowledge representation. Based on personal preferences, a researcher may find one or the other of the approaches more plausible and decide to use the USL system in a particular way. Yet, Smetacek expressed certain preferences during our discussions and this allows me to guess which of the two branches of semantics are closer in his views to USL. First of all, Smetacek does assume that the lexicon will sooner or later include encyclopedic knowledge.⁵ This alone is a strong indicator pointing towards the representational approach of cognitive semantics. Secondly, throughout the discussions with Smetacek, it became clear that SEMAN should be a system for content analysis. With automated ways of analysis of textual data, we disclose indicators of what is happening in the reality – the analyst provides a certain mental model, knowledge representation through the language of USL, and based on this model harvests data about texts. Through such data, one can indirectly deduce what is happening in the real world – but only indirectly. Because the texts are not mirrors of the psychological or objective reality. They mirror what people think about the reality.

⁵ Even if we may question its feasibility.

On its own, components of meaning can be obtained by semantic features analysis – objects are characterized by their features. Yet there is no definite prescription on how to select features. Classic theories in semantics (in the tradition of Alfred Tarski and Donald Davidson) would tend to speak about necessary and sufficient conditions, and propositional functions. This highly analytical approach takes advantage of the apparatus developed for logic, but its binary nature (false, or true) is limiting. It ignores the very presence of a person and assumes there exists one universal truth to which everybody can subscribe; or against which sentences may be evaluated. Meanwhile cognitive semantic theories are typically built on the argument that lexical meaning is conceptual. That is to say, meaning is not necessarily a reference to an entity or relation in some real or possible world. Instead, meaning corresponds with a concept held in the mind of an individual understanding. And as such, it is described as best as the individual can describe it – at the given moment. Later on it changes.

Smetacek is very close to this view and perhaps opposed to classical truth theories of meaning. “Meaning in model-theoretic semantics is a mathematical construct, not a mental object, and is completely independent of use and of speakers...model-theoretic semantics is utterly indifferent to questions of psychological realism, and is not interested in speakers' actual processes of linguistic understanding and production. Furthermore, meaning constructed in this fashion is totally objective: the truth conditions with which it is identified are objective characteristics of the world which do not depend in any way on our recognition of them or even on the existence of a mind to think them.” (Violi 2001, 41)

Yet cognitive semantics will pay attention to processes inside the speaker. Even if not by using means of introspection, but by analysis of usage. The evidence comes from the language use, the utterances are external signs of internal structures. The theoretists construct theory of meaning, and can verify the concept on the actual utterances. Semantics is thus a tool that may serve in this endeavour, it helps to analyse patterns in language use. “Once it is accepted that a theory of understanding is not only part of semantics, but actually coincides with semantic description, given that to explain how we understand is to explain how we mean, there are a number of important methodological consequences.” (Violi 2001, 29)

II.3.2 Semantic fields

The concept of semantic fields can be traced back to Humboldt (1836) who is first credited with the linguistic relativity theory, i.e. with the understanding that different languages encode linguistic and cognitive categories differently. They influence the way people think about reality. The language is therefore no more a passive tool and more attention is given to study of relationships between concepts and how they are expressed. This is then the direct predecessor of the theory of semantic fields by Jost Trier. He showed development and various modifications of a conceptual field in medieval German over the course of time. (Harden 1983, 46)

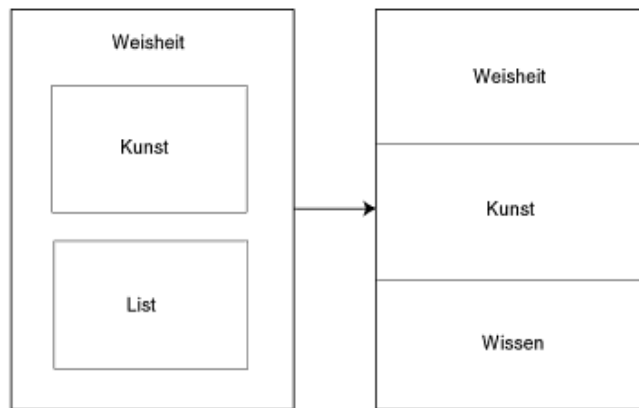


Fig. 2.1. The INTELLECTUAL field's structure in German at around 1200 AD (left) and at around 1300 AD (right)

Illustration 4: The change of the INTELLECTUAL field's structure in German at around 1200 AD (left) and at around 1300 AD (right). Originally appeared in Strapparava, p. 15

Trier applied his analysis to lexical transformations of the medieval German in the “knowledge” domain – he discovered that around 1200 AD the word *Kunst* was used in the context of chivalry and gallantry. The word *List* was used for skills needed outside the court. And the word *Weisheit* was used as a hypernym that encompassed both domains. But looking at the usage one hundred years later, Trier discovered the semantic change. *Weisheit* was no longer used as a hypernym of *Kunst* and *List*, but possessed meaning of religious and mystical experience. *Kunst* attained modern meaning of “artistic knowledge” and a new word *Wissen* appeared. It denoted more general usage of “knowing something” - having knowledge. The word *List* was not used any longer so frequently and usage of other related terms started to reflect the new lexical configuration – and new organisation of the real world. Meaning which was associated with different terms got gradually shifted towards another term and the original word started to be used in a different context, with different meaning. While the lexical form may remain constant, meaning associated with it may in fact change constantly over time.

Trier defined semantic fields as sets of all lexemes connected at syntagmatic and paradigmatic levels (or in only one of them). Thus it is nothing else than a structured subset of the lexicon.⁶ Syntagmatic relations have to do with rules of discourse, ordering of linguistic elements, and on the syntagmatic plane the term acquires value in relation to what precedes or follows it. Outside of this syntagmatic system we have to deal with paradigmatic (associative) relations. They are not linear but substitutive (therefore associative). While syntagmatic relations are manifested in language, paradigmatic relations are “virtual”. They exist in heads of speakers, and if we want to speak about them, we must choose filters. These relations almost presuppose conceptual and mental structures. By assuming this methodology, the theory of semantic fields effectively re-introduces the psychological dimension of the language system. Something, which is in direct opposition to denotational linguistics we discussed previously.

⁶ The distinction between syntagmatic and paradigmatic relations dates back to Ferdinand de Saussure, and corresponds to a dichotomy of linguistic and cognitive domains. One being directly observable through the language and its rules, the other intuitively felt present, but not manifested through rules of grammar.

Another classic example of semantic fields are colours. Colour terms have a precise counterpart in the reality, measurable with the wave length. Terms can be compared across different languages. For those reasons colour terms have been widely favoured for hypothesis testing about conceptualization of reality in different languages. But the situation is very different if semantic fields become more abstract and their meaning must be determined within a given culture and objective verification is often impossible. In such situations it is not clear how to define a semantic field. Frequent criticism of semantic fields then stems from this fact, from their vagueness – it is not possible to clearly and unambiguously define what a particular semantic field means (intention) and what members belong to it (extension). “How can we say that different lexical fields “map” the same conceptual field, that they represent the same conceptual content? What proof is there that the conceptual field is really the same? And what is the nature of this conceptual field? Clearly, it cannot be linguistic, since the terms which describe it differ, but what non-linguistic instruments are there for circumscribing a conceptual area? Definition of the lexical and the conceptual thus seems to run the risk of circularity.” (Violi 2001, 25)

Some authors (Gliozzo and Strapparava 2009) would argue that the linguistic features are sufficient for 'circumscribing' semantic fields as shown in their research. Semantic fields were identified by an algorithm for word clustering, by automated machine-learning procedures. While this approach might be efficient (and eliminates subjectivity), its limitations lay in the fact that it eliminates subjectivity – lexical units are clustered automatically by their closeness and occurrence, but there exist many other ways to group words (because there are so many relations between concepts). Whatever strategy is then chosen, it will always be specific and of limited coverage. From the very definition of semantic fields and their syntagmatic and paradigmatic features it is impossible to ever reach precise definitions in all cases.⁷ There exist too many ways of in partitioning the linguistic and extra-linguistic domain. So it seems unrealistic to ever define everything and once for all.

Semantic fields organize the lexicon internally into defined sets, but these sets will almost always overlap. Boundaries are not be clear-cut and clashes occur. The lexicon is not simply a list but a multi-dimensionally structured set of thematically connected subsets. “In other words, the overall semantic universe of a language can be conceived as a set of interconnected, reciprocally activated semantic microuniverses. This idea has been confirmed in terms of the mental organization of the lexicon by a great many psycholinguistic tests, and it is at the basis of many cognitively inspired semantic theories, including Fillmore's frame semantics.” (Violi 2001, 27)

Semantic fields are thus not representatives of a complete semantic theory of meaning. They are more akin to a discovery tool. One can decide to turn attention to deep conceptual structures (the Greimasian ones, for instance, if one believes in existence of such structures) or one can decide to include only cultural and domain specific features – that is what content analysts do when they prepare coding schemes and select relevant keywords. Semantic fields thus represent not a theory of meaning, but 'only' an important methodological approach. USL and semantic fields are alike, but they should not be confounded with theory of meaning. In fact, particular theory of meaning, vision of the

⁷ And it might be interesting to note, that whilst the first version of the semantic field was based on the automatic word clustering, the most recent version of the Domain WordNet has been prepared manually. I.e. the lexical terms were clustered into domain by humans, using extra-linguistic knowledge. This approach is the same as SEMAN does.

world, or any innate assumptions might be encoded using USL – the formal treatment will be the same, but there is no guarantee of objectivity in the formal treatment of any subjective belief.

II.3.3 Semantic primes and universals

Some people believe that basic universal components of thinking exist and that they are similar (if not identical) across different natural languages. The most famous proponent of this idea is Anna Wierzbicka who started research into semantic primes and universals in the early seventies and which continues until now. (Goddard 1994; Wierzbicka 1996)

The development of the idea of semantic primes is in many aspects similar to that of the Smetacek's *semes* – he devised a list of 300 basic *semes* which were meant to define 'all' other complex concepts. From the initial proposed set of primitives, during iterative clean up, redefinition and (re)discovery of new concepts, research gradually identified new candidates for primitives, so that after 30 years, the list of core *semes* contained close to 2 000 items. This is a very similar development to Wierzbicka's primes. And it is instructive to picture *semes* against *semantic primes* research, even if careful comparison reveals also a lot of differences.

Wierzbicka started her research in 1972 with the first tentative set of 14 primitives – such as: want, do not want, something, someone, I, you, world and a few others. Unlike USL, the primitives were combined using a syntax of a natural language, even if a limited one. Here is an example of a (more recent) definition of a concept *lie*:

```
X lied to Y =
X said something to Y
X knew it was not true
X said it because X wanted Y to think it was true
[people would say: if someone does this, it is bad]
```

And here another example comparing sad and distressed:

```
Sad (e.g. X feels sad)
X feels something
sometimes a person thinks something like this:
  something bad happened
  if I didn't know that it happened
    I would say: I do not want it to happen
  I do not say this now
  because I know: I can't do anything
because of this, this person feels something bad
X feels something like this
```

```
Distressed
X feels something
sometimes a person thinks something like this:
  something bad is happening to me now
  I do not want this
  because of this, I want to do something
  I do not know what I can do
  I want someone to do something
```

because of this, this person feels something
X feels something like this

The search for primitives grew also into the search for universal syntactic patterns (universally valid combinations of primitives) and later on into a pursuit for a fully scaled *natural semantic metalanguage*. The theory became more pronounced and a range of domains, where the theory was applied, was growing as well. Wierzbicka and her collaborators analysed more than 30 world languages and collected an impressive amount of data.⁸

Semantic primes are defined by their 'un-breakable' meaning:

The elements which can be used to define the meaning of words (or any other meanings) cannot be defined themselves; rather, they must be accepted as “indefinibilia”, that is, as semantic primes, in terms of which all complex meanings can be coherently represented. (Wierzbicka 1996, 10)

In this definition we meet the philosophical tradition of ancient Greece with the ideas of enlightenment, particularly that of Descartes and Leibniz. But on the opposite side stand the genius of Ludwig Wittgenstein and the dominant figure of modern linguistics Noam Chomsky. Both of whom refuted the notion of any universals in meaning.⁹ Ideas of basic semantic concepts were embraced and refuted throughout history, and sometimes by the very same people who advanced them. As was the case of the young Wittgenstein, a fervour advocate of formal logic of a language before rejecting it in favour of family resemblances and language games.

The issue of semantic primes is indeed controversial one and so far nobody succeeded in finding the list of truly universal primes:

If at this point, we had to evaluate the work on primitives as developed in the dictionary semantics, we would have to conclude that it has been a failure. There are two major problems. Firstly, no one has been able to convincingly determine an exhaustive list of primitive terms; the attempts often differ considerably, a sign there is no intuitive agreement about how many and what kind of terms there should be. In short, there does not seem to be any *correct* set of primitive terms... a satisfactory list has not yet been found, and above all that all other lexical meanings cannot be derived from them. This, in fact, is the second and more serious problem, which appears insurmountable. However vast the group of selected primitives, it can never thoroughly account for the meanings of the terms not derivable from it. The semantic system cannot be reduced in this way, and primitives do not have an adequate descriptive-explanatory capacity. The result is that primitives are neither really finite, nor comprehensive and exhaustive, nor ultimate atoms... nor do they always avoid circularity. (Violi 2001, 77-78)

⁸ Though for reasons that will be discussed later, semantic primes are not a widely accepted theory in the linguistic domains; “Since 1970s, general enthusiasm for semantic primitives has subsided” (Goddard 1994, 10)

⁹ Even if Chomsky is the most prominent figure of universalism in the grammar and the universal deep structures (mentalist hypothesis) he was opposed to universalism in semantics.

But proponents of the theory argue that it is a known fact that dictionary entries contain limited set of words.¹⁰ That translation into foreign languages is possible. And therefore, there must exist identical concepts across cultures. They will point to the very necessity of having basic, structural items of meaning:

“It is clear that there are words which cannot be defined; and if nature hadn't provided for this by giving all people the same idea all our expressions would be obscure; but in fact we can use those words with the same confidence and certainty as if they had been explained in the clearest possible way; because nature itself has given us, without additional words, an understanding of them better than what our art could give through our explanations.” (Wierzbicka 1996, 12)¹¹

Wierzbicka continues that for Pascal and Leibniz there was never a question of “choosing” some arbitrary set of primitives. What mattered was the choice of concepts that were clear on their own and which could be used to explain other concepts. There is a certain expectation in this claim. That it is possible to arrive at a limited set of semantic primes. Our life-long experience or the fact that we are able to understand different languages points to the idea that fundamental human concepts are somehow innate, being part of the way we relate to and think about the world. Knowledge which is perhaps not conditioned by culture, race, education and social status.

If we look at the main principles of the *semantic primes* theory, we can recognize similarities between USL and *semantic primes*. In fact the only main differences are in the chosen representational system – natural language syntax in case of semantic primes¹² – and in the strong emphasis on lexicalisation. But in principle, USL complies with the first 5 points of the Wierzbicka's programme, as adapted from (Goddard 1994, 8-14)

1. Semiotic principle: A sign cannot be reduced to or analysed into any combination of things which are not themselves signs.
2. Principle of Discrete and Exhaustive Analysis. Any complex meanings can be decomposed into a combination of discrete other meanings, without circularity and without residue.
3. Semantic Primitives Principle. There exists a finite set of undecomposable meanings with the elementary syntax and form of 'simple propositions'

10 Oxford Advanced Learner dictionary (2003) for instance lists three thousand basic terms, only those words are used in definitions of an entry to explain a meaning. And quite often, the definition of the basic entry will reveal some sort of circularity such as when a person is defined as *living being*, *being* is defined as *creature*, and *creature* contains two definitions, one of which says it is a *person*.

11 Citing: Pascal, Blaise. *De l'esprit géométrique et de l'art de persuader*. In *Oeuvres complètes*. Ed. Chevalier. Paris: Gallimard, 575-604.

12 This difference is profound, of course. While Wierzbicka is expressing higher-order logical constructs, USL can define only propositions of existential-conjunctive logic. In Wierzbicka case, the representation is not formalized. “In fact, meanings are very complex structures, built not directly from simple elements such as 'someone', 'want', or 'this', but from structured components such as 'I want something', 'this is good', or 'you did something bad'...” “Concrete” nouns (i.e. names of natural or cultural kinds) will usually exhibit a more static semantic structure, but here too, many different components are usually involved, and these components refer not only to certain inherent features of the referents, but also to the “external frames”-such as habitat, behaviour or typical interaction with people in the case of animals, or the typical situation of use in the case of artefacts.” (Wierzbicka 1996, 171)

4. Natural Language Principle. Semantic primitives and their elementary syntax exist as a minimal subset of ordinary natural language.
5. Expressive Equivalence of NSMs. Any simple proposition expressed in language L1 will be expressible in a NSM based on other language L2 etc.
6. Isomorphism of NSM. The simple propositions expressed in L1, L2 etc will be fundamentally isomorphic (resembling each other)
7. Strong Lexicalisation Hypothesis. Every semantically primitive meaning can be expressed through a distinct word, morpheme or fixed phrase in every language.

Speaking pragmatically, almost every automated system that deals with knowledge representation is working with something that is operatively defined as a *prime* or in other words, *primitives* or *primitive terms*. In fact, Wierzbicka embodies the extreme case of the definition-based approach to the lexical semantics. Speaking metaphorically, the enemy camp is laid just on the other side of the river bank. It is the theory of Rosch, inspired by Wittgenstein family resemblances. (Wittgenstein 1998, 67-77) As Kecskés explains, linguists are divided into invisible and controversial groups, one *semiologic*, which looks at words in their isolation and the way their meaning is manipulated, the other, *onomasiologic*, which focuses on concepts and from there on the multiple expressions that embody them. (Kecskés 2003, 30)

USL represents the latter approach, because the main governing principle is the meaning but not the word or its neighbours. But we shall also not forget the limitations of the lexical semantics. The approach of Wierzbicka, and to some extent of Smetacek, goes one step further and presupposes existence of universal primitives – common to every language (in the first case), or to human race (in the case of the latter). While for Wierzbicka, this is the tenet of her theory, for Smetacek such a claim is a theoretical possibility – something, that should be refuted or studied and bears influence on the way the thesaurus is constructed. But in essence this is not different from the attitude of knowledge engineers who accept some primitives because they are useful for the job. Usability and applicability are then the final criteria for construction (or discovery) of primitives.

“Leibniz also saw clearly the dilemma stemming from the mutual dependence between our knowledge of simple concepts and our understanding of complex ones: to understand complex concepts we have to decompose them into what we assume are simple concepts; but to discover which concepts can be reasonably regarded as the simple ones we have to experiment with many different candidates, checking their power to “generate” complex concepts.” (Wierzbicka 1996, 213)

II.4 USL AND LEXICAL SEMANTICS

USL was conceived as a language for a description of the meaning – more precisely, for definition of features that are the most important for the analysis (notwithstanding the fact that they may or may not constitute all necessary and sufficient conditions to characterize the concept). This approach to the problem is that of lexical semantics, and we shall discuss and delimit areas of interest that are solvable in this context.

The central point to the USL is the **lexicon** – a store of lexical items with explicitly expressed/encoded semantics and very often also encyclopedic knowledge. The semantic lexicon was perceived as an open system and Smetacek insists that it is never fixed and never finished. In the same way as the mental lexicon of a speaker whose knowledge store is not static and who is continuously learns and keeps **forgetting** words. Perhaps this is the main reason why Smetacek did not see advantages in a complex computational representation of the lexicon. And insisted that the representation remains very simple and trivial. So that also the changes and updates to the lexicon are propagated in a trivial and instantaneous manner.¹³

USL is a tool for lexical semantics and certain assumptions accompany such a view. Firstly, lexical semantics is concerned with the meaning of words and word groups which is in quite sharp contrast to the meaning of sentences. Sentence semantics is often described as compositional. I.e. the meaning is extracted from the sentence as being a composite of the individual words' meanings, but of course it is more complex than that. With the USL and lexical semantics we are able to study only words and words groups without taking into account their context. From this view USL is not concerned with the grammar and on its own cannot take advantage of the rules. Words are seen in isolation without the linguistic knowledge.¹⁴ Thus the computer system is not capable of operations that involve fine-grained, but sometimes also very simple distinctions. To give a simple example:

Mary saw a saw.

There is no way for the system to distinguish between *saw* as an act of seeing (in the past) and the *saw* as an object for cutting without the extra linguistic knowledge or without a statistical model of a language (which says what meaning is most likely present at the given position, based on the context). It is clear that from the word alone it is not possible to disambiguate the meaning and the system must be prepared for such variants. In the worst case scenario the lexicon would contain several ambiguous entries:

saw = object instrument cutting
saw = seeing

¹³ Note, this is not the issue of computational complexity or available technologies. In fact, the lexicon is stored in a relational database, with foreign keys constraints and possibly triggers that update contents automatically. Yet the problem of technical implementation was always seen as secondary. The key principal issue is the relative simplicity of the knowledge representation. Thus also the dictionary, viewed from the human perspective has a very simple structure. It can be argued that this is not the easiest and the most effective representation, as my experience showed, but in many occasions is proved to be sufficient. The power of the simple representation was very often realized months after one already worked with the system. A more complex and more powerful mechanism would also prove as more time- and energy-consuming.

¹⁴ In Machine Learning literature, this is the prominent approach, very often denoted as 'bag of words'.

The system would identify twice the same word and assigned to it two different meanings.

This problem can be mitigated by several ways. Firstly, the construction of the dictionary makes it possible to see tokens in context of other words. With the increasing number of tokens their ambiguity decreases (but does their frequency). But that is not everything, the system can employ a number of NLP procedures in the pre-processing stages and translation happens only after specialized linguistic operations were completed. Thus the system can be tailored to be either very fast and produce rather 'noisy' results, or be very careful. Complexity and processing times increase exponentially for certain problems and in some cases this extra work is justified, in other cases, it is not needed.

II.4.1 Words as lexical items

The construction of the semantic dictionary follows the design of the original SEMAN system and we will discuss a few peculiarities in this place.

In the classical linguistic traditions words of the lexicon are listed in their basic entry form, *lemma*. In many computer dictionaries the notion of lemma is taken further to *stem*, i.e. the basic inflexive form of the word – its non-linguistic root. In the USL words are listed in the form of quasi-roots which are basically word stems. However in cases of ambiguous readings, the longest non-ambiguous wording is preferred, often being the full form of the word. For example words *waiter* and *waiting*, though being both nouns, will have the same stem 'wait' in the lexicon. They could be both represented with the forms 'wait' that results in ambiguous definitions being retrieved when one of the form is encountered in the text. But if the lexicon contains a more complete version of the word, for example 'waiter', then most ambiguous parses are resolved.

This behaviour is due to the combinatory principle of the algorithm¹⁵ which constructs all possible inflectional and derivational variants of words.¹⁶ In the present embodiment the algorithm is able to retrieve combinations of prefixes, word stems and affixes that match the form found in the text, and the form with the longest stem is in majority of cases the correct, unambiguous parse of the word. If more than one parse of the word is identified, the system allows for a detailed disambiguation procedures to be put in place which will select the correct lexical form. Empirically, this form of parsing is very fast and accuracy depends on the quality of the lexicon. It is thus relatively easy to improve accuracy with diagnostic information. And the system assists users in finding them.

¹⁵ For details of the algorithm, see V.1 The pattern matching mechanism, p. 115

¹⁶ Derivational affixes are used to create (derive) new words from the stems, very often they do have meaningful interpretation (open-ness). This is in contrast to inflectional affixes which express the grammatical categories such as aspect, case, modality, number, person, tense, voice.

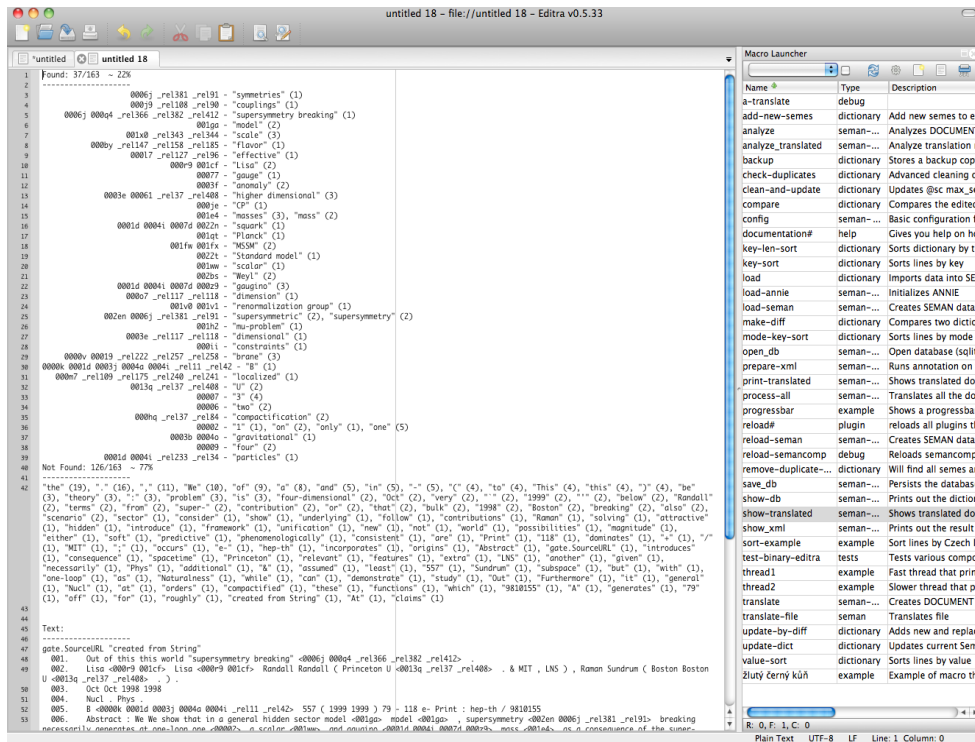


Illustration 5: Example document analysis, showing details of how the document was parsed - with the absolute frequencies of tokens and more details (down the page; not visible).

It is up to dictionary maintainers to decide whether derivational suffixes should have an impact on the meaning of identified words in later stages of analysis. For example, whether the difference between 'was waiting' and 'waited' is an important feature and translated meanings of the two contain different semantic codes. Or, if it is not the case, they shall be assigned the same code and differences considered as irrelevant. The system must allow for a very flexible mechanism that can deal with similar tasks. It will not be possible to define conditions in the form of a dictionary and for those reasons SEMAN contains a simple, but powerful mechanism for definition of conditional workflow routines. Each set of problems can thus be analysed and resolved by separate modules.¹⁷

II.4.2 But what is a word? And how do we define its meaning?

It is a known fact that different languages package meaning into different words, constructs, and syntactical categories. It is quite impossible to say that words and phrases have one, or on contrary do not have one meaning (when is meaning expressed by one word and when by several?). Smetacek approaches this problem pragmatically. It is noted that external forms of words are rather irrelevant, what is central here is meaning – if meaning of a word or group of words is considered equal, for the given application, it should receive the same definition. Trivial as this might seem, it is important to note that the semantic dictionary (in its original form) contains no meaningful labels at place of

¹⁷ For details, see: IV.4.6 SEMAN GUI and scripting control, p. 109

definitions. Instead, one can just see a group of codes. This decision was intentional because it was believed that human readable codes in place of definitions would influence the way the dictionary is built. People would tend to focus on meaning of the individual codes (seen as words), instead of seeking the definition of meaning in its combinatory form.¹⁸

The way the SEMAN dictionary is built forces people to consider meaning first rather than focus on words. The problem is not seen from the lexical side – i.e. we do not concentrate on the form of words in the dictionary. The crucial direction is from meaning towards the form of a word (to the index of meaning). This criterion is applied universally throughout the whole semantic dictionary. If meaning is considered equal (for the given purpose), the definition contains the same number of semantic codes, no less no more. The lexical entry can then be anything. For example:

```
ping fluffy rabbit = id89
Rabbit = id89
```

The word 'Rabbit' is a lexical form assigned same meaning as the first entry. The person who creates the dictionary is responsible to limit grey areas and clean up idiosyncrasies. When she finds that too many tokens are clustered under same meaning, the lexicon allows for a swift and instantaneous changes of definitions.

The reason is that we are not and should not be talking about universal definitions. Definitions are ad-hoc but made consistent, but definitely not acceptable by all. It is a leap of faith, an assumption made to move forward as the whole history of the semantic primitives shows. Some can argue that languages clearly exhibit empirically observable regularities, words tend to occur together, there exist **lexical fields** and by analogy we can find semantic relations. Groups of lexemes belong to a particular activity or area of knowledge. Lexemes exhibit evidence of a network of relations, strongest between members of the lexical field and these are the best candidates for inclusion into the lexicon. But due to number of paradigmatic relations, it will always be possible to see very distant words to share meaning which is crucial to the researched problem.

II.4.3 Words and grammatical categories

In the previous versions of SEMAN, no capability to accommodate different linguistic categories of entries existed. But nouns, adjectives, verbs (open-ended words) need to be given special treatment. For these reasons a special mode signals special meaning and problem-specific logic. Thus we can encode word categories or define our groups. For illustration, we first define categories:

```
e = entities (proper names) - ex. Bill Clinton
q = common noun - ex. dog, woman, house
i = pronouns (immutables) - ex. I, you, we, them
l = logical words - ex. not, and, or, any
```

The form of the entry then distinguishes adjectives from nouns:

```
esq explosive = artifact substance explosion
```

¹⁸ Classification systems such as Dewey Decimal Classification or Universal Decimal Classification have similar approach. Yet they tend to focus on the hierarchical organisation and have very strict rules. For a lay person, unfamiliar codes and combinatory rules make their usage even more difficult.

`esa explosive = attribute explosion`

Then, in the phase of translation the system is instructed to use only a certain subset of entries for lookup; or in post-processing stages only some definitions are retained based on the known linguistic and extra-linguistic features. For example, token `bill` is recognized as a noun, verb and possibly also a proper noun, all of those categories are retrieved for translation. But in the post-processing stage the verb and the proper noun definitions are discarded because part of speech tag will be 'noun' and not 'verb' nor 'proper noun'. For this procedure, it is of course necessary that the grammatical categories are encoded together with the tokens and the part of speech tagging used. For simpler applications such functionality may not be needed and dictionary entries remain simple.

II.4.4 Problematic areas of the thesaurus

II.4.4.1 Homonyms

Homonyms are unrelated senses of the same phonological word. In the context of this work, we are not interested in senses of a spoken word therefore we may ignore homophones. Of importance are only homographs, lexemes that are written in the same form but with disparate meanings. Homonyms can be present in the same grammatical category or outside it, the first case poses considerably more difficulties as we cannot rely on the part of speech information to decide which of the senses is more appropriate. On the other hand, for the second case which covers homonyms outside the grammatical categories, we may rely on the POS taggers that achieved precision of more than 96%.¹⁹

II.4.4.2 Polysemy

When senses of the word are related, lexicographers call this situation polysemy rather than homonymy. Polysemous senses are listed under the same lexicographic entries in dictionaries, but quite often the criteria for senses to be included or excluded rely on intuition or purpose for which the lexicon was created.

school • **noun** 1a an institution for educating children. 1b any institution at which instruction is given in a particular discipline. 2 a department or faculty of a university. 3 a group of artists, philosophers, etc. sharing similar ideas, methods, or style.

school • **verb** 1 formal or N. Amer. send to school; educate. 2 train in a particular skill or activity. 3 Riding train (a horse) on the flat or over fences.

¹⁹ Accuracies of close to 96-97% are typically achieved in English for taggers that were trained on a corpus of at least 10^6 words. (Dale, Moisl, and Somers 2000, 408) Of course POS taggers produce different results based on the quality of the training corpus (if the POS tagger is statistical), therefore there may exist huge discrepancies in the accuracy of the POS and thus the accuracy of the homonym identification. But in general, the inter-category synonymy will be a much more complex problem.

Distinctions are not always clear (and one could argue if they ever could be). The approach of Smetacek to the problem has always been consistent with traditions of lexicographers if possible. Yet definitions are constructed with respect to needs of the analysis. Therefore, some entries are superfluous and may be discarded, other entries may be grouped (joined), yet other entries will be given a prominent position and serve as main senses of the dictionary.

II.4.4.3 Synonymy

Synonyms are different phonological words which have the same or very similar meanings. Unfortunately, the true (exact) synonyms are very rare and most often we will face words whose meanings overlap, but not completely. It is again an informed decision of analysts who will allow or deny the same semantic definition to be present for different words. But the form of the dictionary makes this problem almost a non-issue and synonyms are not causing problems.

II.4.4.4 Antonymy

Given the representation of meaning by USL and the absence of the A-box, the current system is not able to deal with antonymy, at least in the sense of the logical negation. This is also a direct result of the expressive power of the existential logic on which SEMAN is constructed. To recall: “EC [existential-conjunctive] logic can represent everything stored in a database, but it cannot represent negations, disjunctions, implications, or universal quantifiers.” (Sowa 2000, 163)

Ability to deal with antonyms must be part of the inference engine, which, as was discussed previously is built on top of the basic ontology representation. It requires rules of inference and reasoning in one way or another. Thus it is not a part of the current work. This is true about simple antonyms (dead/alive), but also about gradable antonyms like: hot – warm – tepid – cool – cold. It is true also about words representing reversed relations such as here/there, ascend/descend, up/down as well as for converse terms like employer/employee, above/below.

Even if some or possibly all those relations might be described with combinations of categories, meaning is not encoded in their combination, but in interpretation. For instance, if employee/employer are defined as:

```
employee = person in-work subordinate  
employer = person in-work superior
```

The inference mechanism is not contained in the definition but has to be added by external interpretation. Therefore meaning of subordinate/superior link needs to be interpreted by given logical rules. And that is outside the scope of the current USL.

II.4.4.5 Hyponymy/hypernymy

Hyponymy is a relation of inclusion and includes meaning of a more general word, e.g. dog and cat are hyponyms of animal. The more general term is *hypernym* and this relation is responsible for most of the links inside the semantic network. USL is particularly well suited to represent the hyponymy/hypernymy relations. If certain entries contain the same elements, we can conclude that they are all coming from the same taxonomic level (sometimes called 'taxonomic sisters'). If codes of certain entries are fully present in the definition of some other entries, then we can know that a term is a hyponym of another term and it is in the hierarchy lower in the tree (or seen from a different angle, the parent hypernym term has all the features of its children).

II.4.4.6 Meronymy

Meronymy is used for a part-whole relationship between lexical items. Meronymy is different from hyponymy in the aspect of transitivity, we can for instance say that a hand has a finger and finger has a hand, but not that a button has a hole and hole has a button. Hole is a meronymy of a button, it makes its part, but is not its necessary condition. This problem is inherently a question of external-world knowledge, “it is conceptually possible to segment an item in countless ways, but only some divisions are coded in the vocabulary of a language.” (Saeed 2009, 70)

Thus it is recommended not to use the meronymy relationships in the lexicon, unless they are needed for the analyzed problem. The same recommendation would be valid if we deal with relations of a type member-collection (ship-fleet) or portion-mass (drop of water).

II.4.4.7 Difference between names and types

Proper nouns are important parts of the language processing, any system that is dealing with content analysis must be able to distinguish them from the general, common nouns. In terms of logic, the proper names denote particular individuals, and common nouns denote types or predicates. Proper nouns are always having a unique element even if they share features with other individuals or with other common nouns. For instance:

```
esq small = a1
esq four-legged = a2
esq domestic = a3
esq animal = a4
esq cat = a1 a2 a3 a4
esq proper name = a5
ese Elsie = a1 a2 a3 a4 a5
ese Elza = a1 a2 a3 a4 a5
```

There are two things we shall notice here: firstly, the mode of the entry for Elsie and Elza is different, “ese” stands for “english semantic entity” and is different from the common nouns/types, denoted by “esq” (this difference is driven by the application and was introduced while solving problems of how to identify and extract entities effectively).

What is interesting here is that Smetacek proposed to encode even individuals as composites of types, either because it followed naturally, or perhaps because it was planned that way. It might seem unimportant at first but it is not, because here we deal with a metalanguage. First-order logic that operates in the domain of non-linguistic objects is facing problems when the meta-level is introduced. Such as:

Elsie is a cat.
 Cat is a species.
 Therefore, Elsie is a species.

The problem here is that we are mixing two levels. The proper name denotes the individual and the common noun `cat` is a *type*. The way how this issue is solved in first-order logic is that a type predicate is used (effectively a second-order type, that relates the type of `cat` with the individual `Elsie`). Therefore, some individual of a type `cat` does correctly belong to `species`. The `species` is effectively a second-order relation whose instances are individuals like `Elsie`.

$$(\exists x)(species(x, cat))$$

Smetacek must have been aware of logical relations and distinctions in first- and higher-order logical operations, and recommended use of the predicates inside the definition of entities. Thus `Elsie` is not only a `cat`, but she has a `sem a5` which says `Elsie` is proper name. Reasoning engine, if implemented, can then operate with the external world knowledge and conclude that if `Elsie` contains the type of a proper name (or any similar), `Elsie` is an individual and apply rules associated with individuals. The code “`ehe`” on itself would not be enough.

Though we shall note the notational ambiguity of the current system. There is no way to distinguish first-order from the second-order types – everything is grouped together and the reasoning engine would have to have a knowledge base that marks “individual” as a first order type, and the other predicates as modifiers (higher-order relations). What is of direct importance though, is the recommendation to follow the compositional principle of the USL and put higher-order relations in place of semes, not in place of modifiers.

II.4.5 Word sense disambiguation

Even if lexical semantics limits itself to the level of words and their meanings many difficult tasks remain to be solved. One of the crucial is the identification of a correct sense in case the dictionary contains more definitions for the given token. Let us stop for a while between semantics and pragmatics. As was noted before, semantics is concerned with relations of words to objects they denote, and pragmatics is (roughly speaking) about 'how people use signs to convey meaning'. A sentence like 'The deal is broken.' may have a wide range of meanings depending on context, speaker intentions and recipients' interpretation. In the completely neutral sense, we only know that there was some kind of a contract, presumably a spoken one, between the contractor and the contractee. Based on the change of a state, the contract is no longer deemed valid – we do not know whether by one or by all parties. So far, this interpretation can be obtained only from the possible 'common sense' of meaning for the word 'deal'. As well as what it usually means that 'a deal is broken (ended, prematurely)'. But depending on the context, the very same sentence may have a

very diverse meaning. If in the context, more salient meaning of the 'deal' is 'community, gang' than we can see different meaning if the speaker is a policeman or a member of a rival gang. Or, as given in the nice example by Anderson (as reported by (Saeed 2009, 200-201)), subjects of an experiment were asked to read the story below, and then answer questions about it:

A Prisoner Plans His Escape

Rocky slowly got up from the mat, planning his escape. He hesitated a moment and thought. Things were not going well. What bothered him was being held, especially since the charge against him had been weak. He considered his present situation. The lock that held him was strong, but he thought he could break it.

It was generally agreed that 'Rocky was alone and that he had been arrested by police, and is in prison'. When the very same text, but with a different title was presented, people generally agreed on 'wrestler is held by some kind of a wrestling hold and plans to get out of it'.

A Wrestler in a Tight Corner

Rocky slowly got up from the mat, planning his escape. He hesitated a moment and thought. Things were not going well. What bothered him was being held, especially since the charge against him had been weak. He considered his present situation. The lock that held him was strong, but he thought he could break it.

This example, amongst many others, shows that listeners add their own inferences, their interpretation based on the provided context. And this depends on knowledge provided by the discourse topic – thus, it was found that inside a discourse, even ambiguous words tend to have one meaning (sometimes called as 'one topic per discourse hypothesis').

From the practical viewpoint, it is often necessary to construct a dictionary that is specific for the given domain and the given topic. We may intuitively assume, that even if individual words carry many diverse meanings, the prevalent meaning is consistent with the discourse topic. Thus, the dictionary can deal with ambiguity, with precision of up to 70% as shown by (Yarowsky 1992) or even much higher when the disambiguated words were coming from limited number of domains - for a review see: (Christopher D. Manning and Schuetze 1999, Chpt. 7; Navigli 2009). For the more complex context, more sophisticated methods of word sense disambiguation are sought after. This is nevertheless an AI complete problem and general solutions were not found yet. We will discuss relevant details in the chapter V.2. Semantic ambiguity, p. 124.

II.5 CONCLUDING REMARKS ON USL

In this section, I would like to discuss difficulties and possible shortcomings of the USL, starting perhaps non-conventionally with a personal account of my experiences. I remember that at the first encounter with USL and its ideas, I was very much confused by

the ambiguity, vagueness, lack of rules and certain methodological carelessness. The ambiguity and vagueness stemmed from the very properties of the language, but USL does make them stand out. It does not give us the feeling that once a term is analysed and defined, it is fixed and may be used in other definitions. On contrary, the basic principle of the USL is the flexibility. Smetacek also came to the conclusion that every definition leaks and therefore the system must be open for changes. So, on one hand, the system is ambiguous and vague, on the other hand it is also very flexible – it allows for rapid changes.²⁰

But the lack of rules was confusing. I would have thought that basic principles of the knowledge representation should be stated and rules set out (pragmatically, humans will deviate from rules, but at least there is something, some sort of a fall-back mechanism that is able to resolve problems). But the approach of the USL was again different. It did not offer the pleasant security of the theoretical ground, instead, it invited researchers to go ahead, define as many concepts as were necessary and revise them only when inconsistencies were spotted. This approach was radically different from most of the theoretical recommendations but I have to admit that there were arguments in favour of both. In the case of USL, very little time and energy is dedicated to painstaking clarifications of rules and principles, something that philosophers would consider as complete madness. But on the other hand, it is also little 'crazy' that they spent centuries in disputes about the basic principles of knowledge, epistemology and ontology. USL is here on the other bank and I would characterize its approach as a *pragmatic resignation*. It is assumed that the current definitions are wrong, but it is “the best we currently have, so let's see how well they do in the real application”. USL urges researchers to define concepts in the best-possible way, but what really matters is the internal consistency. If there are errors (and they **are** there), they should be coded consistently – **to err systematically** is thus one of the basic principles of USL.

This might mean that content analysis and the knowledge representation should be prepared only by one individual because that person will not be able to spot its conceptual deficiencies in his or her views of the world. Nevertheless, by following the systematic procedures, all of the deficiencies are (hopefully) encoded systematically and when spotted, also rectified systematically. However, by including more people in the process, we introduce different views of the world and if several persons are responsible for construction of the ontology, their differences will be intermixed. Then it will be impossible to discern them – they might err systematically, but if several people err systematically, it will mean complete chaos.

This was a source of many long discussions with Vladimir Smetacek, in which I argued that the person effectively encoded his or her vision of the world in the dictionary and that it is very difficult to align ontologies that were created by different people, not mentioning different domains. And the answer to my doubts were variations of previous arguments – one person is not able to construct the encyclopedic ontology, be it Aristotle or Dewey, they all left it unfinished. But there exist many ontologies, in a different stage of completeness, like UDC or patent classifications, and it is the best we can get. And pragmatically, something is better than nothing. Smetacek believes²¹ that in ontological chaos, there is an order, that people at the end employ their mental capacities in a very similar manner. That

20 Flexibility is after all, also an important feature of any natural language.

21 Together with many others, including Wierzbicka, Goddard, Jackendoff and finally also Plato

they are able to understand each other, relate to external cultures using some fundamental mechanism of meaning. And this principle will be discernible through the law of big numbers and that we shall try, at least, to detect it. Our theory might be wrong and we do not know if it works, but we also do not know if it does not, until we try.

But to be precise and make justice to clarity, I will have to admit again that a *theory of reference*, in other words *semantics* is very vaguely defined in the case of USL. This vagueness may be the ultimate reason for failure. As results bring nothing very special and extensions to the knowledge representation in a form of some logical reasoning machine are necessary. I tried to dispel some of the concerns by showing that USL represents the EC-logic. Nevertheless people are free to implement ontologies as they want and for instance the difference between first-order and second-order predicates may not be respected; in the way how un-ruleful and flexible USL is, it is possibly the case. Also, we do not have a *theory of truth* and in fact, USL is there just to help discover links between certain facts without defining what they mean. If, in the future, extensions are built, they will have to bring in the missing parts, together with *rules of inference*, richer *vocabulary* and more elaborate *syntax*. In case of USL those parts were not built yet.

III. CONTENT ANALYSIS

Perhaps the most cited definition says that content analysis is interested in finding, "Who says what, in which channel, to whom, and with what effect." (Berelson and Lazarsfeld 1948) It was coined by Berelson, one of the most influential figures of content analysis in the 2nd half of the 20th century, and whilst being short, intuitive and captivating, it is also very broad. Application of such an approach is anything but straightforward – it encompasses domains of linguistics, as well as psychology and cognitive sciences. And even though Berelson was strong proponent of empirical methods in research, his views on content analysis as a method for making valid inferences from the text towards the impact in the future are not universally shared by all – for reviews see (Krippendorff 2004a; Neuendorf 2001)

There are many views on definitions of content analysis, and researchers bring them together with different backgrounds and traditions of their own base domains. Content analysis is thus often viewed from diverse phenomenological standpoints – and this problem is not purely philosophical, as it may seem. As researchers subscribe to varying conceptualization(s) of the world, their views of content analysis are significantly different and that influences their execution of the research. For example, in history of content analysis the first studies of mass communication were purely descriptive, but often with the goal of demonstrating the decline of society or changes in manners. The development of content analysis was initially influenced by tight links to the domain of journalism. The first, widely recognized quantitative content analysis appeared in 1893 and was focused on newspapers bearing the title "Do newspapers now give the news?". It compared the contents of the newspapers of the state New York with special attention to news worthy of attention – it is not surprising, that the criteria of what is 'informative' or moral would vary.

The tight links to the domain of newspapers were not coincidental. The fourth power was gaining in strength and in parallel with that, it was becoming more and more important to know and influence public opinion. To study ways how to measure and even control it in the dynamics of mass communication. The first nation wide content analysis of all the German newspapers was proposed by Max Weber in 1910, just 4 years before the outbreak of the Great War, and even though the research was not carried out, war propaganda of the following years brought a fascinating subject of study which produced numerous analyses (for review of history see: Krippendorff 2004b; Neuendorf 2001). And with the Great Depression in the 30's, the importance of content analysis as a tool even increased because at the time it was argued that mass media was aggravating the impact of the crisis with alarming stories.¹ Increasing number of studies analysed the power and

1 Not dissimilar to the call of US Department of State, that the leak and subsequent publication of secret diplomatic cables puts life of Americans and their collaborators in danger

impact of newspapers on general public opinion, and later on, when new media such as radio and television were introduced, content analysis extended its coverage from text to domains of sound and moving pictures.

Propaganda, a weapon employed massively by the major powers during war times, attracted much attention – content analysis was used during the war, in particular by the Allies to analyse the German news and radio broadcasts in order to gauge the morale of the enemy population, the development of new weapons or even the geographic locations of the future war operations (George 1959). Such development naturally led to the establishment of methodology, and a shift of attention from the purely descriptive content analysis studies to 'predictive assessment'. It was necessary to carefully isolate and measure the importance of certain factors, something which was extremely difficult in the amorphous reality of social sciences and with conflicting signals at hand.

The impact of experiences of the World War II and new impulses from the study of personal traits and subjects' inner worlds as carried out in psychology had shown that content analysis could supply important information, even if in many cases only as supplementary proofs. In fact, it was never used as the only method of inquiry.² Empirical research led to the gradual purification of the theory. The first conference on content analysis was held in 1941 in Chicago and the first methodical publication of Berelson and Lazarsfeld, compiling the basics of the method, was published in 1948.³ Content analysis was accepted by the research community as a full-grown method of inquiry into social reality.

Given its noninvasive nature, and also thanks to its relatively low cost, it became popular in other fields such as anthropology, history or linguistics. But as is often the case with the pendular movements, the massive adoption led not only to the greater popularity of the method, but also to the loosening of the scientific rigour. Content analysis became so popular, that it seemed to be everywhere, and everything was considered to be a kind of content analysis. (Krippendorff 2004b, 10) Berelson cautioned against such exaggerated views, warning that it has never been comparable to techniques of intelligence and as a research method it could supply only additional, auxiliary evidence. That information was to be used in conjunction with other resources. Nevertheless, despite many warnings, content analysis, especially quantitative ones, were considered by many to be somewhat magical – perhaps because it employed statistical methods and the numbers gave impression of hard facts.

With time, such attitudes normalized, and the method is not considered to be magical. It is not trusted to be absolutely right, but it is trusted enough to be the sole method of inquiry – if the used methodology is correct. It is often the main and only method of inquiry in many

(<http://en.wikipedia.org/wiki/Wikileaks>). There always exist power groups with hidden political programme or interests.

- 2 As for example described in Peter Conradi, *Hitler's Piano Player: The Rise and Fall of Ernst Hanfstaengl: Confidant of Hitler, Ally of FDR* (Da Capo Press, 2006). The American army used the services of the deserted Hitler's confidant in order to interpret certain personal traits and wartime operations. Hanfstaengl reports were read by president Wilson, nevertheless, Americans never considered such information to be on par with intelligence information. This is testified by many other resources.
- 3 Later published in a revised version in 1952 as 'Content analysis in Communication research'. (Titscher, Jenner, and Meyer n.d., 56)

studies, but of course the situation has changed considerably in the past 20 years⁴. Firstly, never in the past had researchers access to such abundant sources of information as now. Secondly, mankind is developing tools to process huge amounts of textual and non-textual data alike – what was only envisaged a few decades ago is now slowly becoming a reality. People communicate over the web, without even being aware that their actions are recorded, stored, accessed and analysed simultaneously.

Developments in hardware and networking are mirrored in the fields of natural language processing, information retrieval and artificial intelligence. These areas are already beyond their infancy and bear useful fruits, the disciplines had moved past the unrealistic expectations that computers will 'soon' be able to understand and interpret natural language. The modern generation of researchers see computer systems as very effective tools, nevertheless they are also aware of their limitations. They do not expect computers to suddenly start interpreting the meaning of communicated messages, especially not in the same ways as human brains do – especially because we do not even know how human cognitive facilities work (Bickel, Mandik, and Landreth n.d.; West 2001) However, it is possible to build tools that are able to provide useful insights, no matter if the extracted data is simple in its nature.

Similarly to the domain of computer linguistics and machine translation, content analysis was subject to some exaggerated claims and aspirations. Computers were expected to replace human trained coders (“Assessment and Development of New Methods for the Analysis of Media Content” n.d.; Cuilenberg, Kleinnijenhuis, and de Ridder 1998; Krippendorff 2004b) with the ultimate goal of replacing human intelligence altogether. People hoped it would be possible to design tools that automatically process huge quantities of text with results comparable to human processing. Whilst it is already true and was demonstrated that computers outperform trained human coders in certain tasks (King and Lowe 2003), we are still at the beginning. The tasks in which computers outperform people are simple ones - of recognizing concept, selecting important pieces of information, processing immense quantities of data fast. Though the evolution continues, as seen in the areas of machine learning, in many aspects we are still at the beginning:

Computational linguistics, with its current concern for parsing sentences and disambiguating words and phrases, has made only marginal contributions to computational theories of meaning, largely because it theorizes what is general to language, not what is specific to particular nonlinguistic contexts. Content analysis, in contrast, needs to answer specific questions that have not previously been answered about extratextual phenomena of analysts' concerns... Despite remarkable progress, content analysts can hardly claim to have met the challenges of this new era. The imagined analytical potential is far ahead of what can be done today, fuelling the work of many developers of new analytical tools.(Krippendorff 2004b, 309)

It is clear there is great potential here. Even simple solutions may provide very interesting results, especially when the networking effect is exploited, such as crowd-sourcing.⁵ Yet most authorities in content analysis field still maintain, that the method itself can describe

4 For a substantial list of studies in the past, see: (Berelson 1972a) and for the more recent studies (Neuendorf 2001)

5 Perhaps one significant example for all is the study of Twitter messages which predicts the value of commercial products such as films and is proven to be more accurate than expensive consultants (B. A. Huberman, Romero, and Wu 2009; Szabo and Bernardo A. Huberman 2010)

only message characteristics or relationships between these message characteristics – therefore the scope of conclusions drawn from such data is often limited. It is true, that such information can and should be used in scientific endeavours – science seeks to describe, explain, predict and interpret the reality around us – but it is recognized that there is no **direct** relationship between the source of the message and the intended effect on the audience without a sound theory of how the human mind works.

But such a stance is not extreme, there clearly are relationships and it is important to study them. In the past, behaviourists ignored everything which could not be directly observed, counted, measured, or weighed 'objectively' – and yet the paradigm has changed and was dismissed. It is not necessary to move from one extreme towards another one again. Whilst content analysis still strives to be scientific, certain characteristics of texts (messages in general) are already acceptable for the community. Krippendorff lists a few features of texts that are important for content analysis, and they seem to support such reading of literature (Krippendorff 2004b, 24):

- Text has no meaning without a reader (but its characteristic features can be measured)
- Text does not have "one" precisely definable meaning, but there exists a plethora of possible perspectives and interpretations
- Subject is at each moment aware only of a few of the meanings, the rest are ignored
- The meaning "speaks" to subjects – affects them, carries some potential and has effect on the conglomerate of psychological variables in the background (intellect, emotions, experience etc.)
- Those subjective variables of individuals are probably generalizable, but so far there exists no generally accepted theory of 'human cognition'
- Text has certain meaning in the given context – thus it is possible to delimit one or few interpretations from the sea of other possible readings. "The analyst must, in effect, construct a world in which the texts make sense and can answer the analyst's research questions."
- Content analysis of text creates a specific reading of a text - "inferences" are possible from the text towards the context for which the interpretation was constructed

From the points listed above it is apparent, that the analyst is recognized as an important component of the environment - the research questions and the design of experiments inevitably bear traces of the analysts' personalities. The problems are not only in the way in which they interpret results, but also in the way “how” and “what” questions they ask or do not ask. And since content analysts aspire to create a repeatable and fair measurement, they must limit themselves to manifest content. Krippendorff says, content analysis is only “a research technique for making replicable and valid inferences from texts (or other meaningful matter) to the contexts of their use”. (Krippendorff 2004b, 18)

III.1 MAIN TYPES OF CONTENT ANALYSIS

Neuendorf lists four main types of textual content analysis.

1. *Descriptive content analysis* – In this type of analysis, only the characteristics of the message are taken into account. Researchers are careful not to draw inferences about the studied phenomenon outside the studied domain. The analysis describes only characteristics of the content matter – it can be viewed as a summarization or re-statement of existing content matter. Clearly, this type of analysis can serve as input for other interpretations, but as far as content analysis study is concerned, this is not the goal. Examples include description of the TV series, characteristics of shows aired at certain countries, frequencies of keywords, topics, certain rhetorical expressions.
2. *Inferential content analysis* – There is great interest for those wanting to go beyond the description of data, most often in studies of mass-communication. Researchers are naturally keen to discover what is happening in reality, but because reality can only be studied indirectly using products of human communication, it is tempting to draw certain conclusions from these signals. The prevalent view of the communication in the form: *source* → *message* → *channel* → *receiver* seems to support the idea that after characteristics of the message were discovered, one can **indirectly** infer facts about the source or recipients of the message. I.e. what was the motivation of the source to communicate in that way, what effects such a message had on recipients, how to change the message so that desired results are obtained next time. Quite clearly, if such an interpretation is not backed by data from other sources (validated) researchers are in danger of creating wild guesses or unsustainable theories.
3. *Psychometric content analysis* – This type of content analysis is applied in psychology. The method seeks (a) to provide a clinical diagnosis for an individual through analysis of messages generated by that individual or (b) to measure a psychological trait or state of the individual using the messages. Though this method seems to go beyond the characteristics of the manifest content, it involves a careful process of validation in which the analysis is linked with other diagnostic methods. (For review and bibliography see Gottschalk 1997; “PCAD 2000” n.d.)
4. *Predictive content analysis* – This type of content analysis has as its primary goal the prediction of some outcome or effect of the message. Researchers aims to predict receiver or audience responses, for example (Phillips 1979, 1983) has examined the incidence of suicides after newspaper reports of suicides, and the occurrence of deaths due to car accidents following soap opera suicides. Although this type of research is criticized because causal links are hard to prove, especially in behavioural and social sciences, strong correlation can sometimes be shown.

In Smetacek's view, SEMAN is a tool for predictive analysis. However in my opinion that is more wishful thinking than reality. The nature of USL does not make it predisposed for any inferential work, unless there is an A-box built on top of an existing T-box and the nature of USL makes it best suited for descriptive (classical) content analysis. The idea of semes might indeed be useful for the discovery of new relations, as we will see later, and it

indeed works differently than the classical content analysis tools. However, the novelty in the approach does not necessarily make the tool more suited for inferential content analysis. It would be difficult to argue that SEMAN is something more than a new tool for traditional content analysis.

III.2 THE COMPONENTS OF CONTENT ANALYSIS

Content analysis is a scientific method and thus follows verified and repeatable procedures which we will summarize in this section with the goal of highlighting certain elements and problems relevant specifically to the task of the computer assisted content analysis.

As a research method content analysis is consistent with the standards of survey research; an attempt is made to measure certain variables as they occur in the manifest content – its accessible representation. The procedure of quantitative content analysis (summarized by White and Marsh 2006) generally consists of these steps:

1. Establishment of hypothesis or hypotheses
2. Identification of appropriate data (text or other communicative material)
3. Determination of the sampling method and sampling unit
4. Sampling
5. Identification of the data collection unit and units of analysis
6. Establishment of a coding scheme (for hypothesis testing)
7. Coding of data
8. Verification of coding reliability, adjustment of the coding process if necessary
9. Analysis of the coded data, statistical testing
10. Interpretation of results

The process, as outlined above seems linear, but often the contrary is true – especially if the exploratory analysis is done first, or when researchers subscribe to a certain scientific school or paradigm, as is the case of grounded theory, the school of thought prevalent in qualitative content analysis circles⁶. However, for the purposes of this discussion, the generalization of the process as outlined above is sufficient.

⁶ Important concepts of grounded theory are categories, codes and codings. The research principle behind grounded theory is neither inductive nor deductive, it might be characterized as explorative – this leads to a research practice where data sampling, data analysis and theory development are not seen as distinct and disjunct, but as different steps to be repeated until one can describe and explain the phenomenon that is to be researched. The stopping point is reached when new data does not change the emerging theory anymore, until that point is reached, the research is iterative.

In the case of quantitative content analysis, the hypotheses should be tested and falsified. A certain theory of communication provides researchers with a view on the cultural or social phenomena and the task is not the formulation of the explanation of such a movement, but identification and extraction of indicators that confirm or falsify such a description. It is still necessary to identify the variables that support or testify for the presence or absence of an ongoing process, but theory also serves as a starting point. It provides a ready explanation of a studied phenomena, and content analysis is a method to obtain data.

In the first phase, researchers clarify their research questions. A working hypotheses are generated. They will serve as a search spotlight for the next steps. Based on the research questions, **researchers** must select *variables (indicators)* of the studied phenomena. But as there are many ways to look at the definition of meaning, there are also very many ways to study people when they use, express, and interpret the meaning written in the text. A few of the points thus deserve separate treatment.

III.2.1 Coding and categories

In quantitative content analysis, a fairly exact and unambiguous definition of sought concepts is needed for the coding phase. This is called “operationalisation of concepts”. For most part, the operationalisation of the concepts has the form of a coding dictionary or coding scheme (when human encoders are used) and content analysis tools are specifically built to make the coding reliable, fast, and unambiguous. From our viewpoint, the coding is the crucial phase and the tools differ in the way they handle it and the dictionary development.

Indicators from the text are usually grouped under categories – harvested into 'bins' – and these will represent the extracted data, the raw material for the analysis. It is important how fairly the coding scheme captures the studied phenomena. Categories must not be too general, because that would conflate too many signals into one bin, but categories also must not be too narrow, as each observed feature would, in extreme cases, have its own category and that is just as unhelpful. Categories should not overlap, as the signals would be counted several times. Categories must be both unique, and exhaustive, not leaving out important indicators. And because it is not easy to build such a sufficient coding scheme, the process can take many iterations.

The development of the coding scheme is the most laborious and expensive operation of most advanced content analysis studies. It is reported, that such work takes more than 5 person-years of coding (Schrodt 2009, 19) , and the absence of a good coding scheme is effectively the same problem as the infamous 'knowledge bottleneck' of the knowledge representation systems.⁷ Classical content analysis stands and falls with the coding dictionary, especially in case of traditional automated content analysis.

The proper design of content analysis tool needs to ensure:

⁷ And the final reason, why the modern systems of information extraction are focused rather on unsupervised methods, statistical learning and artificial intelligence algorithms. To build a knowledge representation with human experts is often prohibitively complex and expensive (Sowa 2000; Turmo, Ageno, and Català 2006)

1. consistency of coding
2. consistency of categories throughout the times

Consistency of coding is of paramount importance and therefore it is convenient if content analysis tool contains procedures for dictionary evaluation and coding scheme performance measurement.⁸ Various functions for checking consistency, distribution of gathered signals across categories with the possibility of splitting certain parts of the coding scheme in an automated way. And procedures that automatically check integrity of the dictionary.

The second requirement of consistent categories throughout the times is sometimes more a theoretical than a practical possibility. Most of content analysis studies devise elaborate coding schemes, but none of them can expect them to be complete – at least not until a period of prolonged testing is over. If after that the research needs to be replicated, it is necessary that the coding scheme be fixed – or at least versioned. People will need to change, evaluate and revise it – if not in later stages, certainly during the development stage. And small changes will often have profound, unforeseen consequences. Therefore the systems must provide feedback, evaluate results of the coding schema changes and measure performance gains or losses.

Generally, it is beneficial if existing coding schemas are reused – it not only allows for a direct comparison of research results, but also mitigates the knowledge acquisition problem. There exist numerous dictionaries – for example several coding schemes developed for the analysis of armed conflicts in political sciences such as WEIS, PANDA, or datasets with the associated coding schemes such as CIA Factbook. Also the dictionaries with Osgood differential, and Harrold Lasswell's attitude dictionary. These were designed specifically for the purposes of content analysis, but more dictionaries usable for analysis are available from different resources – for example generic, upper ontologies such as Wordnet⁹ or CYC, or domain specific ontologies published in the form of SKOS ontologies such as the dictionary for High Energy Physics. We shall not forget also the more traditional classification systems, such as the Dewey Decimal Classification or Patent Classification which are also successfully used for the task of content analysis.¹⁰

It is often beneficial to choose existing dictionaries, but very often the development of a specialized version will be necessary. Content analysis tools usually make it possible to import different coding schemes and provide tools for dictionary maintenance. It is usually possible to develop a new dictionary, but also convert existing dictionaries into the data structure of a particular system. For example SEMAN has such abilities and they were tested on a semantic network as complex as Wordnet or the HEP taxonomy expressed in

8 Interestingly, when such tools were made available they were parts of the academic coding tools and not the commercial software suites, as will be apparent later on.

9 WordNet is the most widely used ontology for natural language processing. It was started by George Miller and his colleagues in 1995 and it contains 155,287 English words classified by concept types called synsets (synonym sets). They are further split into a hierarchy of types and subtypes with a few other relations, but with no axioms or formal definitions. Wordnet is thus effectively a semantic network of concepts expressed in English language. CYC on the other hand, is a real ontology with the inferential engine suitable for the AI robots. It has the aspiration of becoming a knowledge base for the computer driven human-like reasoning.

10 The possibilities are numerous, for example De Wever reviewed 'only' 15 coding schemes targetted specifically at the coding of the computer supported collaborative learning (the collaborative discussion activities). (De Wever, Schellens, Valcke, and Van Keer 2006)

SKOS format. But at this point it is not possible to compare the tools yet. Without discussing their working mechanisms. Suffice to say that some tools allow us to build a semantic map, and again some others rely on a simple, flat list of patterns. This is an important detail because the internal knowledge representation of the dictionary in the final stages limits the capabilities of the tool.

The limited options of some content analysis tools may not always be perceived as a disadvantage. Because the more complex concepts we try to capture or express, the more difficult it is to select and maintain consistent coding of patterns. There is an imminent paradox of what is possible and what is attainable with the resources at hand – it may be nice to express very complex concepts in a language of USL or any other coding system, but if the final purpose is the simple recognition of patterns, the advantages do not outweigh added complexity. And such complexity will have a detrimental effect on the reliability of the dictionary. Suddenly we are working with a knowledge representation system and if there are more people expressing their views of the world, they will inevitably end up mixing different views of the same world into one representation system.

The differing views will be encoded into one. Sometimes it will be necessary, because simpler coding allow only simple operations – but there lays inevitably a tradeoff between expressiveness and complexity. If we enforce simplicity, we may lose actual meaning. If we enforce expressiveness, we may retain meaning but sacrifice consistency. In the end it is the researcher, who has to take the decision. But the content analysis tool may make such decisions easier when higher expressive power is needed – even if it means more risks. As Berelson said: “[w]hat does it matter that we gain reliability if in the process we lose all our insights?” (Berelson 1972b, 173)

III.2.1.1 Reliability and validity

Reliability means that if different human coders were reading the same text, they should code the same parts of the text with the same categories – i.e. they interpret meaning of the symbols in the same way, consider them to be the same thing and consistently assign the same set of signals to the same category. Results from measurements which are reliable will not change considerably over time. Even if the procedure is applied to a different sample of the same kind, it is expected that results will remain constant despite slight variation. Reliability says that results will not change considerably, that there are no external factors bearing too much influence on the measurement. But it does not say that if we gather wrong data reliably, we will be able to discover something – unless of course by chance.

Krippendorff discusses three types of reliability, together with the accompanying mode of their testing:

Reliability	How to evaluate	How is shown	Importance
stability	test-retest	Important difference in the interpretation of categories	Weak

reproducibility	test-test	As above + discrepancy with other, external studies	Middle
accuracy	test-standard	As above + variations against standard	Strong

Stability means that the process of measurement is not variable through time and that categories are designed in such a way that coders (or the population that was using such codes) understand, interpret and handle them in the same way. Therefore, if the analysis is completed by other researchers, with the same set of tools, they will obtain same results.

Reproducibility is a different facet of the reliability and means, that if the measurement is conducted by other researchers, but using different tools, the results will be still comparable.

The last level of reliability of the measurement is a conformance to the standard. The standard may be hard to obtain in the domain of social sciences. Nevertheless, if available, the variances from it are understood as errors in the measurement, not as errors in the standard.

To achieve high levels of reliability is intrinsically difficult. It was shown, that the overlap even between trained human indexers is on average around 44% (Leininger 2000) which can be corroborated with values reported in (Medelyan 2009; Medelyan and Witten n.d.) which span the range of 13-70% for different thesauri. Therefore for computers this task is very difficult because even people are not sure what is correct.¹¹ The act of coding, when done by people, is further complicated by external factors such as fatigue, distraction and cognitive disparities, cultural backgrounds or education.

These reasons ultimately lead to the development of machine coded content analysis, but it took more time to be accepted. This happened only after research had demonstrated that the machine can outperform people in certain tasks (King and Lowe 2003) – at least with well disambiguated dictionaries or routines that make the processing less ambiguous.

As discussed by Schrodtt (Schrodtt 2009, 66) the reliability of content analysis research consists of three components:

- stability: the ability of a coder to consistently assign the same code to a given text
- reproducibility: the ability of different coders to assign the same code to a given text
- accuracy: the ability of a group of coders to conform to a standard, minimum of omitted (missed out) entries, as well as a minimum of false positives

In the case of machine coded data, the situation is easier in some aspects and more difficult in others. For a given set of patterns the stability of machine coding is 100% because the machine will always code the same text in the same manner. This is particularly useful when a time series is being maintained for a number of years. Because the patterns used in coding are explicitly specified in the coding dictionaries rather than dependent on coder

¹¹ Similarly, in the tasks of word sense disambiguation, on the words that have few meanings, the inter-coder agreements oscilated around 95% however for words with many meanings, the agreement between people was on average 70%. (Christopher D. Manning and Schuetze 1999, 233)

training, the same rules can be used 10 years later. Also the machine will be consistent in making the same mistakes and inter-coder reliability becomes a non-issue, at least as far as coding is concerned. However, we come across a knowledge representation problem – as discussed in the previous chapters.

If the dictionary was prepared by several people, it may contain different views of the world inherently encoded in its structure. The good news is that machines are able to code reliably, provided care was taken in the preparation of the dictionary.

In our experience, the reproducibility of machine coding seems comparable to the inter-coder reliability of humans, and a machine is obviously not influenced by the context of an event or by intrinsic political or cultural biases. (The coding dictionaries may reflect biases, but these will be explicit and can be examined by another researcher; the dictionaries are also applied consistently to all actors and in all contexts.) Furthermore, the machine is not subject to coding errors due to fatigue or boredom, and, once a coding vocabulary has been developed, it does not require retraining. (Schrodt and Gerner n.d., 2:66)

The results reported in King are impressive.¹² Nevertheless it should be also noted, that they apply to a limited domain of problems and that coding schemes are carefully crafted. For example, the language of the source document is simple. There are not many problems with coreferences, a domain specific rather than domain-universal dictionary is used, the coding scheme is well tested and was cleared up by many iterations. A lot of precautions were put in place to fight against pollutants, distortions and biases so that dictionaries are unambiguous and the machine is able to recognize patterns in the text with a high degree of accuracy. The task of content analysis tool is therefore to make such customization not only possible, but also easy.

Validity

Contrary to “reliability”, validity is the name for the “intended concept match”. Data can be valid only if the tools that are measuring them are capable of such measurements. In the particular case of content analysis, the tools are not the software packages, but the coding schemes – these are the real tools of measurement. Validity cannot be estimated by repeating a survey or experiment with the same tool, or by duplicating the data or by calculations that dependent on the previously obtained values. Validity can be estimated only by comparison with other external sources of information obtained independently of the conducted measurements. Estimates can then say how much information obtained in the process is a valid indicator for the studied phenomenon.

In the case of textual content analysis, the nature of data is peculiar. Texts themselves can be considered as indirect representation of reality, but these pictures of the reality are skewed in many different ways – and they can also be interpreted in different ways. Even if we assume that there exists one shared, and perhaps objectively describable reality, it is hard to concede that these 'snapshots' of (one) reality represent it. That the observed change in the data corresponds to the (unobserved) change in the underlying reality and therefore the world of 'texts' is parallel to the changes in the other worlds.

¹² And can be corroborated by the TABARI project which reports 75-85% accuracy. (Schrodt 2009, 7-8)

We could see content analysis in the same light as sociological research and distinguish between different types of analysis levels:

1. description of the manifest content (descriptive studies)
2. content analysis that supply data about the existence of certain patterns (i.e. theory-supporting data extraction)
3. content analysis which interprets (therefore somehow 'understands') the meaning, says how people read and comprehend, how this understanding influences their behaviour

Especially the last type of analysis cannot be conducted without a firm theory of communication and the human mind and without a significant input of the humans, thus it is very difficult to achieve validity from the methodological point of view: (Krippendorf 2004a, 318)

(a) if content analysis serves the purpose of predicting future events and states, it is not possible to validate the results against results of other measurements

(b) the second, more significant case, is when researchers have the external evidence and this evidence is used for the design of the current Content Analysis study. Such results may influence researchers to focus on certain facets and thus the study will be effectively confirming the previous study. Even though formally external, the study will not be independent from the first.

Validity is hardly a problem in a simple study, because people can generally agree that categories measure what they are intended to measure. "In cases where there is high agreement on the definitions of the relevant categories, there is little difficulty in achieving validity in content analysis data". (Neuendorf 2001, 169) But because validity is mainly a problem of definition, and several definitions can be made of a certain category, it is not always easy to make sure that the tool is really measuring what it should. Also the relation between validity and usefulness is contradictory. It is possible to conduct very valid and accurate research, but also very limited and without any value, without taking any risks. Or to take the other route, take risks and validate the data against an external source of information. But content analysis coding tools will not be of any great help here, that is a project design problem.

III.2.2 Data

Content analysis tools can help mostly with the processing of data. It is the responsibility of researchers to make sure that information coming from the conceptualization directly relates to studied questions. The initial decisions in content analysis research have to deal with the question "which units of analysis are important?" – for the purpose of the research, a unit may be defined 'freely' based on the purpose it serves. But it is important to distinguish whether it is (a) a sampling unit or (b) is used as a base for measuring variables, and whether it (c) serves the purposes of reporting final results. Content analysis tools thus have to deal with several data units during the analysis, and be flexible enough to allow for their separation.

Sampling units

They denote WHAT will be included in the analysis. Contrary to census or sampling for the purposes of quality testing, sampling units of content analysis are rarely used as units for reporting. The sampling units serve as 'containers' or pools, from which data is extracted. For example, they can be constructed as issues of a journal, random samples or articles from a given year, all newsgroup messages for the first week of every month (time periods) etc. Obviously they represent the input for the coding phase and must fairly represent the population of all the messages. Content analysis tools should be flexible enough to process various input formats and allow researchers to extract and work with only the important parts of the message in the next phase. Yet in most cases, the problem of sampling is external to the tool used, and is internal to the chosen methodology.

Recording units

Also called units of data collection, these represent 'the smallest possible and meaningful units of the message that contain information about the researched problem'. For example, in psychological content analysis the verbal *clause* is often the best unit of data collection, because it allows one to assess presence or absence of many key markers of a personality. For purposes of the analysis, the word would be too isolated and sentences or paragraphs too broad. The decision depends on the problem at hand. Obvious tradeoffs in the economy of research remain (Berelson 1972b, Chpt 2), but as far as text content analysis systems are concerned, it should be possible to work with individual words, sentences, paragraphs, pages or other units. Tools will necessarily vary in their capabilities to recognize and work with different recording units. But recording units are not what distinguishes one tool from another – it is ultimately the next data unit and its processing which is the most important one.

Units of analysis

These units tell different systems apart. In our work we focus on content analysis tools that use dictionaries, and are specifically suited for frequency analysis of categories – represented by (groups of) words or more generic patterns. Units of analysis are simply 'things being counted', and while they might be identical with the units of recording, in most cases they will represent simpler items. For example category WEALTH will count occurrences of words such as *money*, *dollar*, *pound*, or complex patterns such as *he invested in or company x acquired y*. The task of the most content analysis applications is to assist with the definition of such categories and their content, find such units, or help to find them, and then produce output that can be analysed. Either by the tool itself, or more often, using external, specialised statistical software packages. The units of analysis ultimately correspond to the raw material which will be melted and recast during analysis inside the (external) packages.

Contextual units

There exists another interesting unit type, while not strictly being a 'data unit', it may be very important. Contextual units are simply containers for information that describe the way in which the units of analysis were obtained – simply because for analysts it may be vitally important to know the context in which recording units were extracted, how they were extracted, using what procedures. Also, what was the presumed target audience of the original messages, when the sampling units were selected, how the recording units were chosen and similar. All this data may help interpretation of extracted information,

especially in later stages, when analysts already forgot all the intricate details of the coding. If content analysis tool can record contextual units (presumably many of them in an automated way, creating logs of operations), they will serve the same purpose as workbooks of experiments in the natural sciences.

IV. SOFTWARE FOR CONTENT ANALYSIS

Prevalent classification of content analysis software is based on the difference between qualitative and quantitative content analysis and also on differences between different types of analysis (as outlined in the previous chapter). The two main groups of software are: packages for qualitative content analysis, called CAQDAS (Computer Assisted Qualitative Data Analysis). The other group is composed of quantitative content analysis tools. (For review and extensive bibliography see Koenig n.d.) While it is true that many qualitative content analysis programs now offers modules for quantitative content analysis, there are still many differences in the way in which the research is organized. For example, most of the tools which are of importance and interest to us, work with specially crafted dictionaries, provide mapping operations and the results of the analysis are available as input for more sophisticated processing – usually employing statistical packages such as SPSS, MatLab or analytical tools in Excel. They usually lack the capabilities and graphic interfaces of the qualitative tools, unless they are parts of much bigger software suites.

Lejeune observed (Lejeune 2008, 2009) that content analysis tools can be classified into 3 main categories. Those that:

1. generate classification categories automatically
2. leave the construction of the categories completely to the user, but help with the analysis
3. assist with the analysis using registers

The first group is the subject of research in the fields of artificial intelligence and machine learning. It encompasses a plethora of algorithms, from the simple extraction of concepts based on probability metrics, up to the classification of data using neural networks, statistical learning and similar - for reviews, see (Sebastiani 2002; Dale, Moisl, and Somers 2000; Christopher D. Manning and Schuetze 1999; Sampson 2003)

The second group of tools consist of already mentioned CAQDAS packages (“Assessment and Development of New Methods for the Analysis of Media Content” n.d.; for reviews see: MacMillan 2005)The most visible packages include software like NVivo and MegaPutter.¹ While it is no longer true that they 'only' help to organise the content and the researcher does all the work manually², it is true they were constructed primarily to assist humans to organise content matter. The researcher does the work “manually”and it is her

1 NVivo can be found at http://www.qsrinternational.com/products_nvivo.aspx [Accessed: 24-07-2011] and MegaPutter at: <http://www.megaputter.com/> [Accessed: 24-07-2011]

input that the software helps to manage. This group is the biggest, and usually when the general public hears about content analysis tools, it is software from this category that is used.

Finally, the third group of tools is specially constructed around a dictionary (called a register) and as opposed to the CAQDAS tools, that almost all subscribe to grounded theory, tools in this category show much wider theoretical and methodological variety. Here the analysis is not completely in hands of researchers (like in 2) or completely in 'hands' computer (like 1), but we see various degrees of combination. The registers are often based on the established theory for which dictionaries were built by groups of researchers. For example the Laswell's dictionary for measuring levels of positive/negative interest. The fact that often a standard register is employed of course does not preclude the option of creating a completely new register.

For a comparison, we can cite another example of how content analysis tools are distributed:

Software for content analysis divides, according to its intended function, into three major categories. The first set of programs perform dictionary-based content analysis. They have the 'basic handful' of text analysis functions, involving word counting, sorting, and simple statistical tests. The basic handful are described in the next section. The second set contains development environments. These programs are designed to partially automate the construction of dictionaries, grammars, and other text analysis tools, rather than being analyzers themselves. Development environments are more similar to high-level text-specific programming languages than to freestanding content analysis packages. The third category contains annotation aids. While an annotation aid can often perform some automatic content analysis, it is intended more as an electronic version of the set of marginal notes, cross-references and notepad jottings that a researcher will generate when analyzing a set of texts by hand. (Lowe 2003, 1)

The quote from Lowe succinctly describes the world of content analysis tools. The first category belongs to the tools of classical content analysis, the second more to the world of the new algorithmic approaches and, finally, the third speaks about the CAQDAS tools. By classical content analysis we mean the tradition of examining word frequencies, creating concordances, and building content dictionaries in order to operationalize interesting aspects of document meaning. Of course, also other traditions of content analysis exist. E.g. discourse analysis, cognitive mapping, and collocational clustering, with specialized software available for application of each method. However, SEMAN belongs to traditional content analysis and will be described as such. While it might be interesting to compare SEMAN with CAQDAS tools, as they are amongst the most utilized packages, they were not subject of our work. Our purpose was to focus on the first and the second category of programmes – tools that can help to code and provide basic analytical functions.

The main difference of SEMAN against the existing tools, is that we wanted to provide a solution crafted for USL. But also flexible enough to serve in the more classical paradigm of content analysis. It is somewhat a mix between the two approaches but details will be discussed later. First comes a review of content analysis tools features and then a review of the selected packages. The review may serve as a baseline for comparison with capabilities of SEMAN.

2 Sometimes described as reflexive, instead of “manual” content analysis.

As described by Lowe, tools from classical content analysis category, contain certain functionality. That includes counting frequencies of words and its basic analysis, also summarization and visualisations of results. Word frequency analysis provides a list of (all) the words that occur in a text and the number of times they occur, or a normalized form as a function of word counts and text length. More sophisticated methods split the text into subparts, e.g. chapters, and create frequency lists for each, taking advantage of the fact that certain parts of the text such as introduction or conclusions have somewhat higher information value. The knowledge of the document structure is often used in the parsing tasks, or during information extraction. (Medelyan 2009)

Lists with frequency information can be compared either visually, or using a statistical test to see if certain locations contain significantly more mentions of particular words in one part than another. Another common use of the frequency statistics is to compare treatment of one subject in different sources – to see how different their treatment of it is on the basis of the sorts of words they use.

Statistically this procedure can sometimes be reasonable because the counts from one source are compared with the total counts for all words over all the sources; significant differences may then track differences of emphasis across sources. Some packages make use of synonym lists or lemmatize before the analysis in order to merge word counts. Lemmatization removes the grammatical structure from the surface form of a word, leaving only the stem; words are then counted as identical when they share a stem. For example, a lemmatizing frequency count would treat ‘steal’ and ‘stole’ as the same word. Lists of lemma and synonyms are naturally language specific. (Lowe 2003, 2)

However, the most important functionality is the category frequency analysis. Almost all tools work with a dictionary that allows mapping of certain words into a predefined subset of codes. Category counts provide a slightly more sophisticated analysis. The implicit model of text generation implies the author of the text thought in terms of categories and natural language is used as a medium of transport when the message is recorded. If the content analyst can recover or reconstruct the word set used by the author, the dictionary can be used to decode other texts translating and normalizing their form so that direct comparison is possible.

This is the main usage of the coding tool in content analysis routine where computers are appraised for their speed, and reproducibility of errors. And for the mapping operation, computers outperform humans even for context sensitive problems. (Medelyan 2009; Schrod 2009, 66) Errors in the coding are systematic and to a large extent correctable – while the computer will always make the same error, human coders frequently introduce bias that is hard to spot and even harder to correct. The two factors, speed and consistency, are therefore the main factors that drive the introduction of content analysis tools.

The disadvantage of machine coding lies in complexity of the language, computers do not do the best job in parsing complex and often idiosyncratic expressions that are easy for a normal human reader. It is also substantially harder to provide a computer system that is able to infer facts from the signals, especially if the inference spans over larger areas of texts, as often is the case with texts and their interpretation. Thus content analysis systems may differ in (parsing) speed and ability to recognize complex or simple structures, which will influence accuracy of the coding and the ability of the system to extract useful information. The existence of the dictionary is nevertheless a very important feature of

almost all tools, because it allows researchers to specify a mental model of the world, even if simplified one. The dictionary is used as a mapping from the wild and ambiguous domain of the natural language into a more ordinate form of formal codes with known properties.

The category of software tools that analyse texts using registers is not small. The sites that register content analysis tools³ contain an impressive number of packages. But looking closely, we will discover that almost half of these tools are CAQDAS, such as T-LAB⁴, Wordcruncher⁵ – the other big group of tools specializes in automatic content analysis, without the underlying conceptualization in the form of a coding scheme, such as: Concordance⁶, Crawdad⁷, Hamlet II⁸, Leximancer⁹ and many others. And the packages, that use coding schemes and are built specifically for classical text analysis are relatively few. Interestingly, the newest one was created in 2006, while the oldest was created 40 years ago.¹⁰ Incomplete list contains:

- Diction
- General Inquirer
- Intext
- PCAD2000
- PROTAN
- TABARI
- TEXTPACK
- VBPro
- Wordstat
- Yoshikoder

3 The main sites are: <http://www.textanalysis.info/>, http://www.restore.ac.uk/lboro/resources/analysis/ca_software.php, <http://www.content-analysis.de/software/quantitative-analysis> and <http://academic.csuohio.edu/kneuendorf/content/cpuca/ccap.htm> (dated but still widely referenced)

4 <http://www.tlab.it/>

5 <http://www.hlanalysis.com/WordCruncher/WC.aspx>

6 <http://www.concordancesoftware.co.uk/>

7 <http://www.crawdadttech.com/> Crawdad uses Centering Resonance Analysis, or CRA, a method that applies natural language processing to create a network model of text. Word influence is calculated based on the structural position of the word within the CRA Network.

8 <http://apb.newmdsx.com/hamlet2.html> – software by Alan Brier counts individual and joint word frequencies, the resulting similarities matrix can be analysed using different methods (cluster analysis, MDS).

9 <https://www.leximancer.com/> – a text mining tool that automatically identifies key themes, concepts and ideas from unstructured text.

10 Naturally, it was rewritten several times so that it is compatible with the recent hardware and operating systems.

For purposes of illustration and comparison with SEMAN, I will select a few tools that are similar in functionality: General Inquirer, TABARI, and Yoskikoder. They represent good approximation of the field. For two of them it is possible to obtain the source code, and for all of them the documentation is detailed enough to be sure they work in the similar way as SEMAN. We can leave out many other tools, that are either too small or obsolete, or tools that are parts of much bigger software bundles consisting of many features, which are not directly comparable because they do not deal with coding and mapping of codes onto texts.

Also a number of web services can be found, accessible only via pre-paid subscriptions, but with them it is even harder to see how results were obtained. For this reason we do not include them in the comparison. Certain, such as VRANET¹¹ are widely known to the scientific community as they were used for production of large datasets (King 2003) but their mechanism is hidden behind contractual agreements and available only to departmental agencies and contractors. Therefore the comparison is very difficult. And I also omit the general purpose text engineering frameworks such as UIMA or GATE. These frameworks are used to built almost any kind of text analytical services, but content analysis components constitute a fraction of them. And SEMAN in itself uses these frameworks for its internal operation. My goal is not to list all available tools and content analysis services, rather I would like to describe a few representative tools and compare their operations with SEMAN.

IV.1 GENERAL INQUIRER

General Inquirer is a general purpose system for content analysis and the predecessor of almost any content analysis package available today. It was developed by Philip Stone at the Harvard center of communication and has been used in numerous content analysis studies in the last 40 years. Because of these reasons, I will reproduce here the somewhat fascinating history of its life cycle. This will also serve to illustrate how content analysis evolved together with its computing environment.¹²

Since its original development in the early 1960's, the General Inquirer content analysis software has been adapted again and again to the changing landscape of computing resources. The earlier stages in this adaptation saga are roughly as follows:

- 1) Originally programmed in several languages (COMIT, BALGOL) for the IBM 704-709-7094 mainframe series, the system was reprogrammed in the more powerful PL/I that became available with the large IBM 370 mainframes. These mainframe

¹¹ <http://vranet.com/> [Accessed: 24-07-2011]

¹² There is an interesting parallel with SEMAN, also SEMAN existed in the version for mainframes and was in initial phases limited by memory space and CPU issues. From there we could trace certain design choices.

programs were then supplemented with AUTOCODER programs operating on the smaller, more accessible and hands-on IBM 1401 computer for searching and retrieving text stored on magnetic computer tapes.

- 2) When time-shared resources first became available at MIT's Project Mac and then on commercial time-shared systems in the late 1970's, the PL/I General Inquirer programs were adapted to supporting real-time content analyses. Time-shared computing, usually by connecting a teletype or electric typewriter to a mainframe computer over a phone line, enabled us, for example, to perform content analyses of TAT stories as soon as they were typed, giving immediate scores for such categories as "need achievement."
- 2) In the 1970's and 1980's, the General Inquirer might have been adapted for the growingly popular Berkeley UNIX platforms, which at the time ran mainly on DEC and Data General computers. However, IBM mainframes continued to be available and offered continuously improving computing capabilities in speed and available memory size, while the skimpy PL/I versions eventually available on UNIX machines never did facilitate a straightforward implementation of the General Inquirer on a UNIX platform.
- 3) When PC and MAC computers increased their RAM and hard-disk storage capacities enough by the mid-1990's to handle the General Inquirer content-analysis procedures, the Inquirer was reprogrammed for them using the Dartmouth College "TrueBasic" language with its pioneering capabilities in effectively handling very long text strings. A "run-time" PC or Mac compilation of the TrueBasic General Inquirer was distributed that did not require users to have the TrueBasic language on their computers. General Inquirer users could then complete an entire content-analysis project on their personal computer, often also enlisting desktop statistical software packages that also by then had become available on personal computers, such as SPSS and JMP, to complete their analyses.
- 4) By the late 1990's, Java's widespread distribution on personal computers supported large Java programs being run on them. The General Inquirer was therefore reprogrammed in Java for personal computers as well as other platforms. This adaptation of the General Inquirer was made available without cost to academic users by downloading a zipped package containing the Inquirer's Java procedures, its dictionaries, and its disambiguation rules. In addition, a General Inquirer website was created to provide detailed information that might be helpful in using the system.(Stone 2001)

Dr. Stone passed away in 2006 and General Inquirer is now available as a web service usable only for simple projects, but definitely deserves a place in the current section because General Inquirer was the first software of its kind that showed it was possible to conduct content analysis studies using computers and not human coders. Even if in many cases the research was focused on counting the surface features of a text and those features were taken for symptoms of psychological features, the principles of General Inquirer are in use everywhere:

1. It counts features, but also retrieves segments of the text with given characteristics (selected by user). Researchers can change the dictionary based on the given feedback. Improve accuracy.

2. General Inquirer also counts sentences, and displays sentences with given characteristics – it is not bound only to words, recording units are wider.
3. Provides output with statistics on word and sentences. The output can be imported to external statistical packages.
4. General Inquirer is distributed with a prepared dictionary, but is not committed to it. Users can build their own dictionaries.

For the first time, General Inquirer allowed the replacement of the manual coding scheme with a set of computer rules. This practically eliminated the problem of reliability of coding. Computers were coding consistently under the same conditions. This also forced discipline on content analysis researchers as suddenly they were required to formalise rules, provide very exact and precise operational instructions. Such a development had a positive impact on the scientific nature of the method, and finally, the boredom of the study was limited only to the phase of designing the dictionary (which is actually a creative process).¹³

IV.1.1 Disambiguation rules

What makes General Inquirer stand out from the crowd is not only being the first, but also the fact that it was employing extensive routines for the disambiguation of English homographs. Homograph is the word that is written in a same way but based on the context takes very different meanings – such as “kind”. It can be an adjective, but also a noun with many meanings such as 'species', 'type', 'somewhat'.

The disambiguation was accomplished using specially constructed rules, not a general disambiguation approach per se, but considered to be a useful subset of it. The following account describes the details: “Our hope was to endow computer with a limited but useful ability to resolve lexical ambiguity, that is, to discriminate a useful set of senses for *some* high frequency words of English, with a *reasonable* degree of accuracy.” (Shapiro, Tackett, Dawson, and Markoff 1998, 57)¹⁴ The rules, when translated to language, look like this:

To select the appropriate sense of the word “last”:

Consider instances not in root form such as “lasted”, “lasts”, “lastly” and so forth. If the word ends in “ly”, that is “lastly”, assign the sense FINALLY. Otherwise, if the ending is is not “ing” (e.g. “lasted”, “lasts”) assign the sense ENDURE, REMAIN. If it is “ing”, look to see if the following word is tagged as a determiner (such as “a”, “an”, “the”) or a preposition (e.g. “lasting a”, “lasting the”, “lasting until”). In that case, assign the sense ENDURE, REMAIN. Otherwise (no determiner or preposition following) assign ENDURING.

¹³ For more details, see: <http://www.wjh.harvard.edu/~inquirer/kellystone2.htm>

¹⁴ Wuoting: Edward Kelly and Philip Stone. "Computer Recognition of English Word Senses." North-Holland Linguistic Series, 1975.

If the word appears in its root form, that is, simply as “last”, first look to see if the following word is “time” or “fall”. If so, in case the previous word is not “the”, assign the sense PREVIOUS (e.g. “last time”, “last fall”). In case the previous word is “the”, look to see if the word before “the” is a form of the verb “to be” (as in “is the last time”) in which case assign the sense FINAL.

In all, five different meanings of “last” are identified, and eighteen rules, some of which are logically dependent on others, are formulated to distinguish them. This complexity is not at all atypical of the 1.070 dictionary entries for which disambiguation rules were written by 25 collaborators over a period of 7 years. (Shapiro, Tackett, Dawson, and Markoff 1998, 57)

The mentioned period of the last 7 years is the period before the publication of the Stone's book, therefore 1968-1975. In the current state, General Inquirer contains over 13 thousand word roots and 6,336 disambiguation rules. These rule were engineered by skilled programmers in the many years since the first version and the last version of the software from 2006, when the development stopped. The rules were generated from the corpus of the encoded texts that were available to Stone and his team, using KWIC and manual inspection of the context, which means if the feature was not present in the corpus, the case was probably not identified. But as was mentioned earlier, Stone and Kelly aimed only at disambiguation of the “most frequent words with a reasonable level of accuracy.”

IV.1.2 Dictionary

General Inquirer is basically a mapping tool. It maps each text file onto dictionary-supplied categories counts. The current version combines the Harvard IV-4 dictionary content-analysis categories and the Lasswell dictionary content-analysis categories, and five categories based on the social cognition work of Semin and Fiedler, making for **182 categories** in all. The view on the categories might be similar to that of Smetacek's semantic primes (see Table on page 59).

The General Inquirer categories were developed for social-science content-analysis research applications, not for text archiving, automatic text routing, automatic text classification, or other natural-language processing objectives, although they may be relevant to some of them. Many categories were initially created to represent social-science concepts of several grand theories that were prominent at the time the system was first developed, including those of Harold Lasswell, R.F. Bales, Talcott Parsons and Charles Osgood. It also included categories relevant to "middle-range" theories, such as David McClelland's theories regarding needs for achievement, power and affiliation.¹⁵

Each category is a list of words and word senses. A category such as "self references" may contain only a dozen entries, mostly pronouns. Currently, the category "negative" is the largest category with 2291 entries. Users can also add additional categories. An example of

¹⁵ Stone says that normally, each entry in a category is given equal weight. However, this was a choice of the category developer rather than any inherent limitation of the computer. Earlier versions of some Inquirer dictionaries included categories with weightings, but were found to be difficult to obtain agreement about the weights and they added little to the validity counts. Currently, none of the categories employ weightings.

the dictionary below shows several meaning for the word CONTENT, we can also notice comments written after the pipe “|” character, they were probably used by programmers that constructed the disambiguation rules.

```

CONTENT#1  Noun Know | 21% noun: Meaning or constituent elements
CONTENT#2  Noun Object Comnobj | 8% noun: "Contents"--that which
is contained
CONTENT#3  Pos Modif EMOT Pstv Psv Pleasur | 58% adj: Satisfied
CONTENT#4  Pos Modif Pstv Pleasur Psv | 13% adv: "Contentedly"--in
a satisfied
           manner
CONTENT#5  IAV Pos SUPV Pstv Psv Pleasur | 0% verb: To satisfy
6    TOR(K+0,K+0,APLY(4),,LY.
7    TOR(K+1,K+1,APLY(5),,DEF3.DEF2.
8    TOR(K+0,K+0,APLY(3),,ED.
9    TOR(K+0,K+0,APLY(2),,S.
10   TOR(K+0,K+0,APLY(5),,ING.
11   TOR(K-1,K-1,APLY(3),,BE.LY.LINK.VB.DEF1.
12   WOR(K+1,K+1,APLY(3),APLY(1),WITH.TO.

```

The General Inquirer software includes the sense disambiguating procedures as discussed above, but they are not part of the dictionary and cannot be edited or extended by a user. A simplified rules-based POS engine is used to assign the senses and the General Inquirer also cautiously removes common regular suffixes so that one entry in a category can match several inflected word forms. A category entry can be an inflected word (for example, "swimming"), a root word ("swim" would match "swimming", if "swimming" is not a separate entry) or a word sense (for example, "swim#1") identified by the disambiguation routines of an inflected or root word form. These English stemming procedures, integrated with English dictionaries and routines for disambiguating English word senses, limit the Inquirer system to English text applications.

Text files are grouped for processing into folders. The output is a matrix of "tag counts" for each category, with separate rows of counts for each file processed. The main output from the Inquirer is basically raw frequency counts and indexes scaled for document length. Statistical tests outside the program can then be employed to evaluate whether there are statistically reliable differences between the texts or groupings of texts being studied.

Text Statistics

tag	N	%	words
Pos	29	4.73	ADVANCE#2=2 ALLOW#1=3 AUTHORITY=1 CALM#1=1 CALM#3=1 EXCITEMENT=1 FAIR#1=3 FREE#1=3 HOPE#2=1 LIGHT#1=2 MAIN#1=4 OPPORTUNITY=3 REAL#1=1 RECONCILIATION=1 SUPPORT#1=1 VERIFY=1
Pstv	27	4.4	ADVANCE#2=2 ALLOW#1=3 CALM#1=1 CALM#3=1 EXCITEMENT=1 FAIR#1=3 FREE#1=3 HOPE#2=1 LIGHT#1=2 MAIN#1=4 OPPORTUNITY=3 REAL#1=1 SUPPORT#1=1 VERIFY=1
Affil	6	0.98	LEAGUE#1=1 PARTICIPATE#1=1 RESPOND=1 SUPPORT#1=1 US=1 VISIT#2=1
Virtue	14	2.28	DEMOCRACY=3 DEMOCRATIC#1=1 FAIR#1=3 FREE#1=3 OPPORTUNITY=3 REAL#1=1
Pleasur	3	0.49	CALM#1=1 CALM#3=1 EXCITEMENT=1
Neg	21	3.43	BADLY=2 BATTLE#1=1 CRITIC=2 DENY=1 FLAW=2 FORCE#3=1 INEFFECTIVE=1 MANIPULATE=1 MISS#1=3 NERVOUSNESS=1 OPPOSITION=3 PRISONER=1 REFUSE#1=1 STARK=1
Ngtv	17	2.77	BADLY=2 BATTLE#1=1 CRITIC=2 DENY=1 FORCE#3=1 MANIPULATE=1 MISS#1=3 OPPOSITION=3 PRISONER=1 REFUSE#1=1 STARK=1
Hostile	13	2.12	BOYCOTT#2=3 CAST#1=2 CRITIC=2 DENY=1 OPPOSITION=3 REFUSE#1=1 SUBDUE=1
Vice	2	0.33	BADLY=2
Pain	1	0.16	NERVOUSNESS=1
Strng	60	9.79	ADVANCE#2=2 ARMY=2 AUTHORITY=1 BASE#2=1 BATTLE#1=1 CERTAIN#4=1 CONTINUE#1=2 CONTINUE#2=1 CROWD#1=1 EXERCISE#1=1 FORCE#3=1 FREE#1=3 GENERAL#3=1 GOVERNMENT=1 INCLUDE=1 LARGE#1=2 LARGE#3=2 LEAGUE#1=1 LOT#1=1 MAIN#1=4 MANIPULATE=1 MANY#1=2 MILITARY=2 MONITOR=1 MORE=1 MOST#1=1 MUCH=2 OFFICIAL#1=1 ORDER#3=1 PARTY#1=5 POLICE=1 PRESIDENT=2 REGIME=1 REGIMENT=1 RULE#3=1 RULE#4=1 SUBDUE=1 SUPPORT#1=1 SURE#1=5 TROOP=1 WIN#1=2
Power	34	5.55	ALLOW#1=3 AMBASSADOR=2 ARMY=2 AUTHORITY=1 CAMPAIGN#1=1 ELECTION=10 FORCE#3=1 GENERAL#3=1 GOVERNMENT=1 MANIPULATE=1 MAY#1=1 MONITOR=1 OFFICIAL#1=1 ORDER#3=1 POLICE=1 PRESIDENT=2 REGIME=1 RULE#4=1 SUBDUE=1 SUPPORT#1=1
Weak	9	1.47	FLAW=2 MISS#1=3 NERVOUSNESS=1 ONLY#1=1 PRISONER=1 REMOTE=1
Subm	2	0.33	PRISONER=1 RESPOND=1

Illustration 6: An example output from the General Inquirer processing a news from Burma polls. Categories are in the left column, matched words in the right one.

IV.2 TABARI

During the 90s, the panorama of content analysis changed considerably. Firstly, with the development of the rapid news reporting, the focus shifted from newspapers towards news wire services. It was also established by empirical studies that mere machine-assisted coding¹⁶ of news was not more efficient than completely manual coding – thus the energy and funding was redirected towards the development of completely human-independent coding. This move was accompanied by changes in the geo-political area, with the dissolution of one superpower and the end of the Cold War era. The two major enemy camps ceased to exist and the attention had moved to very specific subjects in limited areas. After 1989, suddenly there were no superconflicts, but individual, geographically located threats.¹⁷ The amount of information accessible electronically steadily increased and that made it possible to gather enough 'signals' even for the globally insignificant issues – in contrast to the previous situation of the major competing superpowers. Even small incidents generated a wave of responses (signals) and this information was then used in content analysis. Paradoxically, the wiring of information into the global network made local analysis possible.

In this environment, the project KEDS (Kansas Event Data System) was born at the beginning of 1990s. It benefited from NSF grants and from the second wave of revived interest into the analysis of a large scale data (General Inquirer being a representative of the first wave). The National Science Foundation grant on “Data Development in International Relations” in the early 1990s had a more profound impact:

The Global Event Data System (GEDS) at the University of Maryland grew out of this project, as did the early work on KEDS and several other experimental projects. The DDIR project marked a transition between the DARPA-style event data research and contemporary approaches, and the various articles in Merritt, Zinnes and Muncaster (1993) show a mix of old and new techniques. These methods slowly diffused into the policy community, and by the end of the decade, event data were employed in the development of experimental early warning systems at the U.S. Department of Defense, in the dynamic modeling phase of the State Failures Project, and in Switzerland by the FAST project. (Schrodt and Gerner n.d., 15)

On the technical side, KEDS was written in the Pascal programming language and worked only on Apple Macintosh computers. It was used in the years 1995-2001, but gradually became obsolete and difficult to maintain. It was decided to rewrite it under the name

¹⁶ i.e. the coding in which human coders had programmatic tools to preprocess the text, but it was finally humans that decided about each and every category assignment. Sometimes called as human-assisted coding.

¹⁷ With the global terrorist thread, the situation is changing again – what remains, at least for some areas, is the need to analyse local signals. Locally, but on a global scale.

TABARI (Textual Analysis by Augmented Replacement Instructions) in Spring 2000. The main core of the system is written in the language C++, and TABARI also contains various utilities for text filtering and pre-processing written in Perl and Java. But TABARI, as a successor to KEDS, applies the very same principles and operational modes as KEDS.

IV.2.1 What TABARI does

TABARI extracts *event data* from the stream of text using a pattern recognition techniques. In this context, *event data* is a name for a semi-formalized structure with the following four components:

```
date source target event
```

To give a specific example, taken from (Schrodt and Gerner n.d., 1-4), the report:

```
July 23, 1990: Iraqi newspapers denounced Kuwait's foreign
minister as a U.S. agent Monday
```

corresponds to an event which is in the coding scheme defined as

```
122: "Denounce; denigrate; abuse."
```

In this event, Iraq is the *source* of the action and Kuwait is the *target*. Together, this information generates the event record

```
900723 IRQ KUW 122
```

where 900723 is the date of the event, IRQ is a standard code for Iraq, KUW is the code for Kuwait, and 122 is the assigned category.

The first item is usually the date of reporting – though TABARI allows this date to be shifted by the text itself. For example, if the report contained the words *week ago*, it is possible to recognize this compound tokens as time shifting event and change the date of reporting accordingly. This facility, and the presence of dates in general, is important for time-series analysis.

The most important part of the *event record* is the group of the last three elements which could be freely described as actor-act-object triad. The sources as well as the *targets* are the political entities, such as states, leaders, political parties and similar. The verb then corresponds to the event, the relation between the *source* and the *target*.

TABARI is specifically designed to process short news summaries, usually only the *lead sentence* of such summaries. This detail is not without significance as other similar systems are designed to work on portions or the whole texts. TABARI, however, employs shallow parsing procedures which would fail in many other circumstances, but work well for news wires. Specifically because the journalistic practice dictates that the lead sentences contain the essence of the whole article, in a condensed, and rather predictable form – the source data thus follows certain patterns and grammatical complexities are limited. This allows TABARI to extract event data with a relative high accuracy. The studies comparing the accuracy of the predecessor of TABARI, KEDS against the human encoded data (using the same coding scheme) showed on average a correlation of .80 in the reporting of dyads (source-target elements), and on average .92 correlation in identification of actors (see Illustration 7 (Schrodt and Gerner n.d., Chpt 2, p. 17)).

Table 2.3. Comparisons of Event Data Sets by Actor-Year

KEDS-WEIS			COPDAB-WEIS	
year	by dyad	by actor	year	by actor
1982	0.74	0.95	1966	0.80
1983	0.59	0.77	1967	0.83
1984	0.76	0.90	1968	0.82
1985	0.84	0.90	1969	0.92
1986	0.80	0.92	1970	0.51
1987	0.91	0.98	1971	0.47
1988	0.96	0.99	1972	0.14
1989	0.93	0.95	1973	0.55
1990	0.69	0.90	1974	0.68
1991	0.80	0.94	1975	0.36
			1976	0.76
			1977	0.87
			1978	0.93
Average	0.80	0.92		0.66
N	42	7		128

Illustration 7: Correlation between computer and hand coded data, in the case of first column, the same scheme (WEIS) was used. In the case of the second column, the different coding schemes are used and therefore there is an additional level of discrepancy.

IV.2.2 How TABARI works

TABARI uses a system of syntactical pattern recognition. This mechanism is written in a configuration using special syntax and several dictionaries play an important role:

- **Actors:** proper nouns that identify the political actors. The dictionary is used for naming *targets* as well. In the new version (since 0.7), there exists also a dictionary of *agents* and it lists the possible qualifiers of the actors, such as `prime minister of`.
- **Verbs:** for TABARI the verb is the most important part of a sentence and is used for identification of the event and assignment of the event code to the final dyad¹⁸. The elaborate system of phrase patterns phrases can be specified inside this dictionary. Phrases are then used to distinguish different meanings of a verb – for example PROMISED TO SEND TROOPS has a different meaning than PROMISED TO CONSIDER PROPOSAL and will be classified using different event codes. The dictionary of phrases also provides syntactic information on the location of the source and target within the sentence.

¹⁸ By dyad it is meant the couple of involved entities, a source-target relation. The dyad in this sense is effectively the same as 'predicate' in logic or 'triple' in the parlance of the semantic web.

As a general-purpose coding system, TABARI lets users define its operational rules, or at least change them considerably by simple rewriting of the dictionary – unlike General Inquirer, where the disambiguation rules cannot be changed. All of TABARI's files are stored externally as simple ASCII (“text”) files and can be edited using a word processor. Certain parts of the dictionary, the verbs and actors, as well as their associated codes, can also be interactively edited using the dialog, command-line interface. The editing routine also keeps track of the coder who added each phrase to the dictionary and the date the phrase was added. To illustrate the work of the TABARI system, we can consider this example from the demo actors dictionary.

```
AL_GORE    [USA] ; pas 26 apr 03 TEXT-02
ALBRIGHT   [USAGOV 930206-010114] [---] ;ems 02 Feb 2001
ALBRJGHT   [USAGOV <010114] [---] ;ems 02 Feb 2001
ALBRKGHT   [USAGOV 930206-010114] [---] ;ems 02 Feb 2001
AMROTH_TANKS [AMRTNK]
ANORIEN    [GON]
ARAGORN     [RNG]
ARGENTIN    [ARG]
```

And the verbs dictionary:

```
ABANDON    [345] ;pas 15 Jul 2003
- * EFFORT  [987] ;pas 15 Jul 2003
ACCORD     [---]
- SUGGESTED_+ * DEMAND_{ GENEVA_ | INTERNATIONAL }_CONVENTIONS
[111] ; used to test blank padding of {}
- SUGGESTED_+ * DEMAND_{ GENEVA_ | INTERNATIONAL }_CONVENTIONS
[222]
- INCHING { CLOSER | NEARER } * [083] ;tony 4/29/91
- IN_* [075] ;pas 2 May 2005; pattern CONNECT-3 7/14/03
- IN_* [075A] ;pas 2 May 2005; pattern CONNECT-3 7/14/03
- IN_* [072] ;pas 2 May 2005; pattern CONNECT-2 7/14/03
- TO_* [073]
```

The dictionary is a plain ASCII file, with patterns and special codes. A standard input format is used for the dictionaries. Words are entered in upper case; codes for actors and events are enclosed in square brackets []. If two words must be consecutive, they are connected by an underscore; if the two words are separated by a space, other words can intervene. For example, the text “agreed to provide a loan” can be matched by the pattern AGREE LOAN but not the pattern AGREED_TO_LOAN (we will describe more details about the special syntax later).

The input to TABARI is a file containing a set of sentences, each prefixed with a date and other identifying information, and followed by a blank line. The lines should not exceed 255 characters limit, otherwise they will be truncated. Each new record is delimited by a blank line. The total size of any single text record is limited to a maximum of 2047 characters. TABARI will also apply number of limitations to filter out malformed texts

TABARI Word and Clause Types

Words	
Null	Word has not been classified (not displayed)
Litr	Literal: string of characters that occurs as part of an actor, verb or pattern
Actor	Actor
Verb	Verb
Time	Time-shifting word
Attrib	Attribution word
Determ	Determiner: A_, AN_, THE_
Noun	Explicit noun; verb shifted to noun by determiner or capitalization; and null-coded actors
Adjectv	Adjective
Auxil	Auxiliary verb—WAS_, WERE_, BEEN_—used in passive voice detection
Byword	BY_, used in passive voice detection
Comma	Comma ",", used in subordinate clause detection and compounds
Pronoun	Pronoun: HE_, SHE_, HIM_, HER_, IT_, THEY_, THEM_, THEIR_
Conj	Conjunction: AND_, BUT_, OR_, NOR_
Number	Strings that begin with a digit except when part of a phrase, as in Group of 77
Issue	Word in ISSUES set
NullTag	Type was cancelled because it overlapped with a subordinate clause
Clause tags	
Clause	Clause in compound sentence
Compound	Compound noun phrase
Reference	Pronoun coreference
Subord	Subordinate clause
Replace	Used to standardize actor names in XML input
NullTag	Deactivates tags in subordinate clauses (not displayed)
Halt	End of text (not displayed)

Illustration 8: List of types that TABARI recognizes

(using defaults, which may be changed in the configuration file), such as no coding is done for sentences with less than 8 tokens, or eliminating words which contain more than 63 consecutive characters. Here is an example input:

980216 REUT-0001-01

Egypt's President Hosni Mubarak warned in an interview published on Monday that the situation in the Arab world could deteriorate if the United States attacks Iraq for failing to comply with weapons inspections.

980216 REUT-0002-01

Iraqi Foreign Minister Mohammed Saeed al-Sahaf said on Monday that he was going to Paris only to take a message from President Saddam Hussein to French President Jacques Chirac about Baghdad's showdown with the United States.

980216 REUT-0003-01

Israeli businessmen, Jordanian officials and foreign bankers agreed on Monday that the Israeli-Jordanian peace treaty was not producing economic dividends quickly enough.

These are the lead-sentences extracted from a newswire service reports (TABARI website provides several utilities for processing of several formats of the major news providers, such as Reuters or Factiva). When coding events, TABARI goes through the following steps: (Adapted from Schrodtt and Gerner n.d., Chpt 2)

1. Word classification

The source text is first converted to a standard form. All letters are changed to capitals and commas are delimited with spaces. TABARI then checks each word in the text to see if it occurs in the actor, verb, and agent dictionaries. If the word is found, it is assigned the appropriate type (e.g. actor, verb, pronoun, conjunction etc.); otherwise it is designated as untyped. TABARI disambiguates verbs that can also be nouns (e.g. ATTACK and FORCE) by detecting the presence of the articles A, AN, and THE. It also disambiguates proper names that could be verbs by looking at mid-sentence capitalization. Most of the subsequent parsing operations deal only with the words that have been classified by type. See Illustration 8, on page 64 for the list of all recognized types.

2. Processing local grammatical structures

After the elements are tagged, TABARI will process local grammatical structures. This means that actor identities are assigned to common nouns, also pronoun references are linked with the nouns, and two actor references are translated to a single actor (e.g. Israeli Prime Minister Rabin is reduced to a single reference Israel), compound noun phrases are recognized and subordinate phrases delimited by commas eliminated. If customized "rules" are used — for example a general rule to deal with the English-language passive-voice construction — they are applied at this point (we will see details in the section IV.2.3). Sentences that appear too complex for TABARI to process are written to a separate file rather than coded.

3. Event coding

The program next attempts to match the patterns associated with each verb in the sentence to phrases from the *.verbs* dictionary. As we mentioned earlier, the verb constitutes the crucial element for identification of the event — and also relative to the position of the verb the actors and targets of the action are identified. Each phrase is associated with an event code. Patterns typically distinguish between direct objects, as in the distinction between *promised military aid* and *promised to veto*. If a verb phrase corresponding to an event is identified, the program finds the *source actor* and *target actor* associated with the *verb*. The *source* is usually the first actor in the sentence; the *target* is usually the first actor following the verb, provided that actor has a code distinct from the code of the source. If no such actor is found, the program then looks for an actor prior to the verb that has a code distinct from the code of the source. If the source or target are compound phrases, these are expanded into multiple events. Only the first verb

corresponding to an event is coded, unless the sentence is compound (i.e. contains a conjunction not associated with a compound actor), in which case each clause of the compound sentence is checked for an event.

4. Information output

Following the coding phase (and if TABARI is used in the interactive mode) the main display will show the source text along with its date and identification number, the coded events and some summary statistics. The main display can also show the parts of speech assigned to various words (if configured and supported by the terminal): actors are shown in red, verbs in blue, agents in green, pronouns are replaced with their references and text eliminated by subordinate phrases or null codes is shown crossed-out. This display is useful for quick overview of the text results and refinement of the dictionary. This dictionary development mode is a powerful feature of TABARI because it allows for making interactive changes in rules of coding with results being immediately visible after recoding (which is usually very fast). The whole editing is controlled by a keyboard shortcuts, there is no graphical interface, all operations happen in the terminal window.

In the non-interactive mode the extracted events are simply written to an output file. The coded output can be formatted in a variety of ways, including tab-delimited formats for use in database, spreadsheet or statistics programs.

The default event output format is:

```
<date> \t <source code> \t <target code> \t <event> \t <label>
```

where \t is the tab character. Additional information can be added by TABARI if configured to do so. Example output:

```
040401 ISRSET ISRGOV 220 (FORCE) FORCED
040401 USAGOV ISRGOV 031 (MEET) HOLD MEETINGS 4.4. EVENT FILE 43
040401 ISRGOV USAGOV 031 (MEET) HOLD MEETINGS
040401 JOR SYR 121 (CRITICIZE) SUSPECTS
040401 PALGAZ PAL 094 (CALL FOR) CALLING ON
040401 USAGOV ISR 042 (ENDORSE) ENCOURAGED
040401 BEL NTH 023 (NEUTRAL COMMENT) SAID
040401 SYR USAGOV 171 (UNSPECIF THREAT) STRAINED
```

```

Date: 08 Jan 95                      Record : DEMO-08

The ambassadors of Arnor, Osgiliath and Gondor presented their credentials to
Ithilen's president on Wednesday in a further show of support to his government
by their countries. /*

[Compound actor list]
{ ARN / ITH / 032 } 950108
{ OSG / ITH / 032 } 950108
{ GON / ITH / 032 } 950108 */

Coded events:
950108 ARN    ITH    032    (VISIT) PRESENTED CREDENTIALS
950108 OSG    ITH    032    (VISIT) PRESENTED CREDENTIALS
950108 GON    ITH    032    (VISIT) PRESENTED CREDENTIALS

Select: N)ext  P)arsing M)odify R)ecode A)utocode O)ther Q)uit ->
0 THE          <clause>
1 AMBASSADORS  <determ> THE_
2 OF           <litrl> AMBASSADOR
3             <prep> OF_
4             <compnd>
5 ARNOR        <actor> ARNOR
6 ,            <comma> ,
7 OSGILIATH    <actor> OSGILIATH
8 AND          <conj> AND_
9 GONDOR       <actor> GONDOR
10            </compnd>
11 PRESENTED   <verb> PRESENT
12 THEIR       <prnoun> THEIR_ <refernc = 3 >
13 CREDENTIALS <litrl> CREDENTIALS_
14 TO          <litrl> TO_
15 ITHILEN'S   <actor> ITHILEN
16 PRESIDENT
17 ON          <litrl> ON_
18 WEDNESDAY
19 IN          <prep> IN_
20 A           <determ> A_
21 FURTHER
22 SHOW        <litrl> SHOW_
23 OF          <prep> OF_
24 SUPPORT     <litrl> SUPPORT_
25 TO          <litrl> TO_
26 HIS
27 GOVERNMENT  <litrl> GOVERNMENT_
28 BY          <by_> BY_
29 THEIR       <prnoun> THEIR_ <refernc = 3 >
30 COUNTRIES
31            </clause>
Continue display? (Y/N) ->

```

Illustration 9: An example coding of the diplomatic communication in the 'Kingdom of the Ring', TABARI allows for very quick edits and recoding of the whole corpus

TABARI provides also the statistics on the number of recognized entities from each of the dictionary, for example the following list shows the entities from the actors dictionary that were recognized during the coding:

```

0 ARAGORN
0 ARGENTIN
13 ARNOR
0 ATHENS
0 AXNOR
1 BARAD_DUR

```

TABARI does not contain any statistical procedures for evaluation of the data, it only writes the cumulative category frequencies. This output effectively contains only nominal-level data (categorical), event series must be aggregated to produce a numerical (interval) values suitable for import into the standard statistical programs. There is a special program called KEDS_count used for such purposes. It can produce any CSV separated values and aggregate them into a given output. This utility is very generic and might be used by other projects. It is flexible and allows for a range of output scenarios. The example output is shown in Illustration 10, below.

Date	N	Scale	10	11	14*	5*	17*	77*
860303	2	-0.33	0	0	0	0	0	0
860317	0	0.00	0	0	0	0	0	0
860331	2	-1.20	0	0	0	2	0	0
860414	0	0.00	0	0	0	2	0	0
860428	1	-3.40	0	0	0	2	0	0

Field definitions:

Date	Date of the first day of the period being aggregated (or month and year for monthly aggregations). The format of the date can be modified.
N	Total number of events for the period
Scale	Sum of the scaled codes for the period
remainder	Total number of events of each type of code tabulated

Illustration 10: Example of the aggregation of the categorical scoring scheme (WEIS) into the numerical values coding Scheme (Goldstein)

IV.2.3 Pattern matching

TABARI relies on sparse parsing of sentences – rather than using full syntactical analysis. It will identify actors (mostly proper nouns and possible compound proper nouns) in the vicinity of which the verbs are found, and with help of the verb, the direct objects within the verb phrase are found – or, in the case of passive voice, the entities are swapped at the position of the direct object with the actor. The source of the event is equal to the subject of the sentence, event code is equal to the verb, and the object of the verb is the target. In the old version of KEDS, the success of such a mechanism largely depended on the form of the sentences available for coding because KEDS concentrated on a traditional (at least for English) Subject-Verb-Object structure (SVO). This pattern is very common in the lead sentences and can be used reliably, nevertheless authors of the system were still finding a lot of cases where the default parsing could not deal with a relatively simple, but frequent cases. Thus a new system of pattern matching was invented.

This mechanism distinguishes TABARI from all other systems, making it a very flexible and powerful tool. The possibility to provide your own patterns is very important, even if authors say that the differences in accuracy between the old and new versions are not big (see Illustration 11, below), certainly thanks to the classical SVO structure of the English sentences, which also KEDS handled well. Nevertheless, the patterns are a new and powerful feature that deserves separate discussion.

Table 1
WEIS Coding of 1979-1998 Levant Data

All events	correlation	N KEDS	N TABARI
Verbal cooperation	0.95	54,543	58,259
Material cooperation	0.90	6,208	7,593
Verbal conflict	0.94	13,262	15,296
Material conflict	0.95	18,673	22,512

Table 2
WEIS Coding of 1990-91 Iraq-Kuwait Crisis

Mediator activity ⁷	correlation	N KEDS	N TABARI
Verbal cooperation	0.87	4,818	5,066
Material cooperation	0.74	598	583
Verbal conflict	0.82	950	971
Material conflict	0.85	727	821

Illustration 11: Comparison of KEDS and TABARI system on a large corpus of Levant Data.

When identifying phrases, TABARI looks at the clause – usually the parsing stops when a conjunction or comma is encountered. By default, TABARI checks for standard regular suffixes (-s, -es, -ed, -en, -ing) and the stemming is also activated, for example the basic entry of a verb

KILL

Would match also

KILLS

KILLED

KILLING

However the following form would restrict the match only to the basic form:

KILL_

To define a more complex pattern, TABARI uses the following codes:

- * substitution character for the lead verb
- wildcard meaning 'any group of tokens'
- null code tells the system NOT to ignore the event
- \$ to mark the position of the source
- + to mark position of the actor
- % to mark an entity that is both an actor and the target
- ^ (caret) symbol instructs TABARI to skip some entity (not to use it)

And the software has also the following possibilities for partial matches (wild-card endings) and their consecutive placement:

XXXX YYYY

XXXX can partially match, YYYY does not need to follow immediately

XXXX~_YYYY

XXXX can partially match, YYYY must follow immediately

XXXX_ YYYY	XXXX must match exactly, YYYY does not need to follow immediately
XXXX_YYYY	XXXX must match exactly, YYYY must follow immediately
{X Y Z}	X is synonymous with Y and Z

It is a simple but flexible set of patterns for definition of complex rules. Moreover, the patterns seem minimalistic but it is clear they stem from real world extraction tasks. The following examples illustrate how the patterns are used.¹⁹

```
BREAK_ [1717]
{BROKE_ BROKEN_}
- * TREATY [161]
_ * BONE [182]
```

This pattern would match: BREAK A TREATY, BROKE BONES and similar, but it would NOT match BREAKS TREATY.

Another example:

```
ATTACK [122]
- * CRITICISM OF [042]
- *_HELICOPTERS [- - -]
...
POUND [223]
- BRITISH_* [- - -]
```

Will force TABARI to ignore the event where ATTACK HELICOPTERS are used, as well as BRITISH POUND or BRITISH POUNDS because the category code is empty [---]. At the same time, it will allow for coding of ATTACKED CRITICISM OF as well as ATTACKED FIERCELY CRITICISM OF because by default, TABARI skips intermediate consecutive words.

To write:

```
- * BACK~_POLICY [062]
```

Means that the form BACK can have a wild-card ending, but it must be followed immediately by POLICY, such as BACKS POLICY, BACKED POLICY, BACK POLICY but not BACKED ANOTHER POLICY

Phrases can specify also the location of the *source* and *target* by using the tokens \$ and +:

```
ADVISE
- +_WAS_*_BY_$
```

would recognize the following:

¹⁹ Interesting is the fact that the former version, KEDS, also allowed very complex patterns, but using different rules and syntax. Worth mentioning is the fact that the current system is much simpler.

Egypt was advised by the United States

In this example, the actor locations are used to reverse the default matching rules for identification of the source and target when a verb phrase is in passive voice.

The pattern matching mechanism is indeed a powerful way to define complex processing of the phrases – it is not used strictly to construct formal parsing grammars, perhaps because TABARI is used mostly by social scientists and they would find it too difficult to use such tool. But the patterns remain simple and clear, it is tangible that the authors of the system did not want to burden the work of analysts with too many rules, yet felt the available subset is sufficient enough for all their needs.²⁰

IV.2.4 Dictionary

The coding system that was used in TABARI by its creators, is derived from the World Event/Interaction Survey (WEIS) (McClelland 1999) which was one of the prevalent coding schemes for the political events data, together with another coding scheme COPDAB (Azar 1980) from the early 1970. WEIS and COPDAB coding schemes are general and comprehensive, meaning they strive to cover every possible area in the domain of analysis. But of course more specialized coding schemes exist, often invented by individual researchers, focusing only on specific correlations or subsets of behaviour and TABARI was successfully used with them. (For a detailed bibliography of the dictionary developments, see Schrodtt and Gerner n.d., 13)

While the COPDAB coding system was maintained and expanded by the GEDS project (<http://www.bsos.umd.edu/cidcm/geds/>), WEIS continued to be the most widely employed coding scheme, even if often extended to provide greater detail in the coding of domestic conflict events. The most notable of these WEIS extensions was done by the Protocol for the Analysis of Nonviolent Direct Action (PANDA) project at Harvard in the mid-1990s (<http://data.fas.harvard.edu/cfia/pns/cs/panda.htm>), which produced a global, Reuters based event data set covering 1984 through early 1995. To accommodate domestic events, PANDA more than doubled the number of WEIS categories, while providing a systematic table for translating PANDA codes to WEIS codes. More recently, this effort has been extended to the IDEA coding system, which is designed to be used in the edition of the World Handbook.²¹

The TABARI authors followed suit and developed their new coding scheme called CAMEO -- Conflict and Mediation Event Observations, which borrows from IDEA and offers several new features compared to WEIS:

- New tertiary sub-categories specific to conflict mediation

20 The TABARI codebook manual contains almost 20 pages of examples of the pattern matching rules.

21 IDEA can be found at: <http://www.vranet.com/idea/>

- Substantially expanded the categories for "use of force" which allows for finer granularity in distinguishing levels of violence
- Number of WEIS categories that, in the experience of TABARI authors, could not have been reliably differentiated in machine coding were removed
- A new systematic hierarchical coding scheme for substate actors

This development shows how important role the coding scheme plays. For example datasets produced using the VRA coder are available (King 2003) but not the coding scheme itself.²² Also in the case of TABARI, the coding scheme is not available for download – even if the WEIS categories are available on the site. The coding scheme represent the substantial investment²³, as we could guess from the description of the older KEDS framework:

The KEDS dictionaries we have used to code international events in the Middle East contain about 650 actors and 3500 verb phrases. The PANDA project used slightly larger dictionaries: 880 actors, 4300 verb phrases and 200 agents. Earlier work ... on automated processing of English-language reports of political violence indicated that dictionaries on the order of 5,000 phrases are necessary for relatively complete discrimination between political events, so these KEDS-compatible dictionaries are probably close to having a relatively complete vocabulary. Despite the fact that PANDA is coding the entire world while the Kansas project codes only the Middle East, the actor dictionaries are about the same size because a fairly complete list of actors is required to identify potential targets of events. Dictionaries for coding internal events are somewhat larger, although the number of additional phrases required is usually in the tens or hundreds, not the thousands. (Schrodt and Gerner n.d., 46)

IV.2.5 Concluding remarks

TABARI is one of the systems for the extraction of even data – and perhaps the only open-source available for more than a decade. While it has been available for many years, it is interesting to see that it took many years for analysts to adopt the new paradigm of content analysis. A few reasons are discussed by (Laurance 1990), such as low sensitivity to detection of certain events, distrust of humans for early automated methods of quantitative content analysis and also the technical problems together with low level of user friendliness of the packages.

We find the similar account also in the paper by Schrodt that summarizes the history of TABARI (Schrodt 2006, 5, emphasis added):

TABARI is generally stable and has been used intensively in a number of projects at the University of Kansas Center for International Political Analysis over the past five years. We have periodically added some additional specialized features at the request

²² Albeit online coding manual can be found at:

²³ Or in other words: the real “knowledge bottleneck” of knowledge representation systems.

of external funders, and are slowly eliminating a few remaining quirks, but for the most part are no longer doing active development. Instead, we are producing data sets, which was the reason we got into this project in the first place. **Fifteen years ago.**

The TABARI system is successful and as the citation shows, the authors do not feel pressed for addition of new features. For the type of analysis that TABARI does, especially with the existence of the powerful pattern matching, the system is able to adapt to new requirements. However, it still remains focused on the limited set of the content matter (news clippings) and lacks sophisticated features. Schrodts discussed them in several places (For detailed treatment see: Schrodts and Gerner n.d., Chpt 2, 5), to finally conclude that TABARI would need considerable redesign to take advantage of the new capabilities such as assignment of parts-of-speech, resolution of ambiguities, full sentence parsing, and larger repertoire of sentence forms – all these elements could improve the already high accuracy of the system, but it would also require a lot of changes. Therefore the system does not include them now and probably will not have them even in the future either.

IV.3 YOSHIKODER

Yoshikoder is another tool for content analysis. It was developed by Will Lowe for the Identity project at Harvard university²⁴ and released as open source in 2006. It is written completely in Java. Therefore it can, unlike the previous tools, run on any major operating system and is also not limited to the ASCII input. And as it was designed 16 years after TABARI, it also contains an intuitive graphical user interface.

While at the time when Yoshikoder was developed there already existed many tools for content analysis (the commercial ones also provided a graphical interfaces and many advanced functions), it is an interesting fact that Will Lowe, author of the software, felt the need to develop this new tool. It is even more interesting if we know that Lowe collaborated on many projects that generated content analysis data (King and Lowe 2003; Lowe 2003, 2008, n.d.) and he must have known the field very well.

Nevertheless, something was missing. 1) Firstly, data produced by other tools were not transparent. Before release of Yoshikoder, only TABARI was available as an open-source software and for the other content analysis tools it was not clear what algorithms they used and how the data was aggregated. 2) Another reason was economic; Yoshikoder can provide services comparable to those of commercial tools at much lower costs.

The Yoshikoder is designed to help non-technical social scientists perform classical content analyses on text in arbitrary languages. Using the Yoshikoder helps support the replication standard and annoys people who sell similar functionality in proprietary packages, but it is also part of a larger project to unify, standardize, and disseminate the theory and technology of content analysis. (Lowe n.d.)

24 http://www.wcfia.harvard.edu/conferences/04_initiative_identity/overview [Accessed: 24-7-2011]

Entry	Count	Proportion
Laver and Garry	112	0.092
Laver and Garry > Culture	1	0.001
Laver and Garry > Culture > High	1	0.001
Laver and Garry > Culture > Popular	0	0
Laver and Garry > Culture > Sport	0	0
Laver and Garry > Economy	19	0.016
Laver and Garry > Economy > +State+	2	0.002
Laver and Garry > Economy > -State-	5	0.004
Laver and Garry > Economy > =State=	12	0.01
Laver and Garry > Environment	34	0.028
Laver and Garry > Environment > Con	34	0.028
Laver and Garry > Environment > Pro	0	0
Laver and Garry > Groups	1	0.001
Laver and Garry > Groups > Ethnic	1	0.001
Laver and Garry > Groups > Women	0	0
Laver and Garry > Institutions	43	0.035
Laver and Garry > Institutions > Conserv...	4	0.003
Laver and Garry > Institutions > Neutral	15	0.012
Laver and Garry > Institutions > Radical	24	0.02
Laver and Garry > Law and Order	10	0.008
Laver and Garry > Law and Order > Con...	10	0.008
Laver and Garry > Law and Order > Lib...	0	0
Laver and Garry > Rural	0	0
Laver and Garry > Urban	0	0
Laver and Garry > Values	4	0.003

☒ Show categories only

Export Close

Illustration 12: Dictionary reports for the distribution of categories

IV.3.1 What Yoshikoder does

The basic functionality of Yoshikoder is in statistics, simple counts of words – i.e. counts of tokens that were found in the documents, either in absolute frequencies or proportionally relative to the document size. More interesting functionality is the reporting of the category counts of the coding scheme. The basic dictionary report counts dictionary matches in all the currently selected documents and can display totals for the categories, as well as for individual matching patterns. In the first column, the name of dictionary entry is shown as a path. The second tab shows the same information normalized for document length, as a proportion of words in each documents (see Illustration 12 above).

Yoshikoder also produces concordance reports, that lists all the matched patterns together with their local context for quick visual debugging of the patterns. And also a specific report for comparative purposes. Yoshikoder can compare two documents based on the frequencies of their categories matches by computing the relative risk ratio and also the confidence interval for the obtained values (the details will be described later).

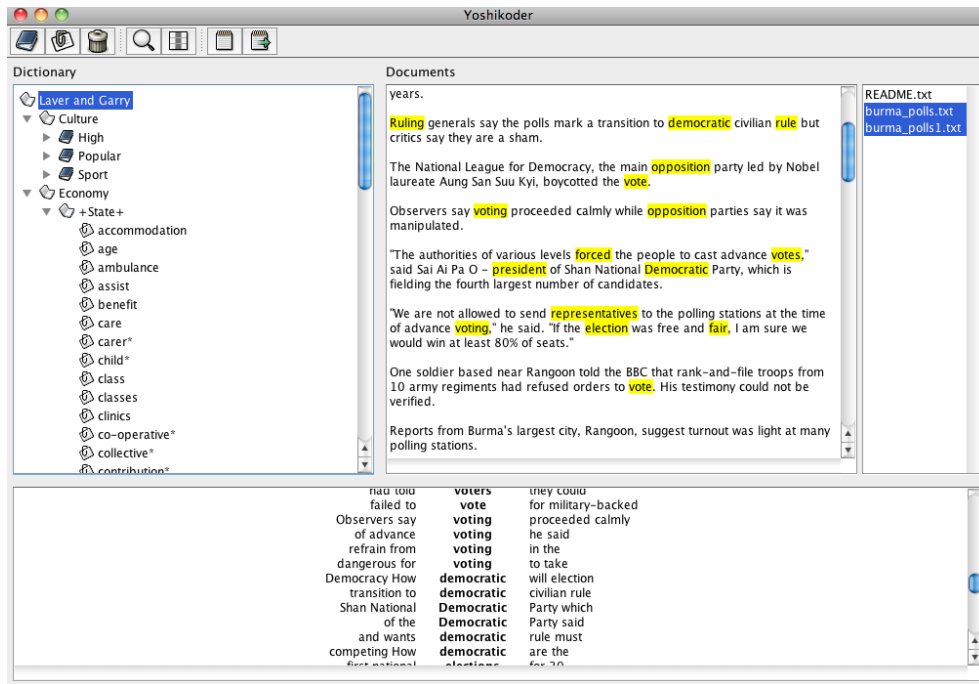


Illustration 13: The interface to Yoshikoder with the visible dictionary tree and the highlighted matches and the concordance report at the bottom.

As is standard with content analysis programs, data can be exported in a variety of formats and analyzed with more powerful statistical tools.

Word frequencies for
(1) burma_polls.txt
(2) burma_polls1.txt

Word	Count (1)	Count (2)	Prop. (1)	Prop. (2)
the	60	78	0.069	0.066
and	7	32	0.008	0.027
in	19	32	0.022	0.027
to	31	27	0.036	0.023
of	25	23	0.029	0.019
a	21	21	0.024	0.018
are	12	19	0.014	0.016
is	8	19	0.009	0.016
for	11	18	0.013	0.015
they	4	18	0.005	0.015
party	6	16	0.007	0.014
that	8	13	0.009	0.011
election	9	12	0.01	0.01
their	3	11	0.003	0.009
elections	5	9	0.006	0.008
it	7	9	0.008	0.008
suu	4	9	0.005	0.008
an	2	8	0.002	0.007
burma	9	8	0.01	0.007
just	0	8	0	0.007
what	0	8	0	0.007
but	6	7	0.007	0.006
by	4	7	0.005	0.006
constitution	0	7	0	0.006
he	3	7	0.003	0.006
i	1	7	0.001	0.006

Export Close

Illustration 14: Unified Frequency Report for two documents

IV.3.2 How Yoshikoder works

Yoshikoder works with the **hierarchical** dictionaries, it recognizes the terminal nodes as patterns and inserts them into categories, the categories are the parental nodes in the dictionary.

Categories represent concepts and the patterns within them represent their indicators that can be observed in text. The GUI allows editing the dictionary entries directly from inside Yoshikoder. Adding category/pattern, removing, dictionary changes are thus very simple and intuitive. It is immediately visible, whether the new pattern matches portions of texts. Dictionary entries are displayed as a tree, with patterns nested under the categories that contain them. It is also possible to rearrange entries just by dragging and dropping them within the tree. Since categories often have hierarchical structure they can be nested in other categories.



Illustration 15: Graphical view of the dictionary inside Yoshikoder and also the xml structure of the data

Pattern matching is simple; the wildcard character '*' allows matching of any sequence of characters (zero or more than one characters). Behind the scenes, Yoshikoder converts the pattern into a regular expression pattern and uses it in searches. Pattern entries thus allows

the matching of single words or sequences of words.²⁵ For example: `chin` matches the word `chin` but not the words `chinese` or `cochin`, `chin*` will match `chin` and `china`, but not `cochin`, and `*chin` will match `chin` and `cochin` but not `china`. The wildcard character can be applied in any part of the pattern name, any number of times. For example, `ch*n*` will match `china`, `chan` and `chinese`.

Categories may also have a description, and the patterns can have associated scores. For example if the dictionary contains a category `Sexual abuse` and inside the category, there are several patterns such as `molest*`, `rape*`, `sexual assault*`, each of these patterns may have a different weight associated with them, if say `molestation` was being seen as more serious than `rape` (for the given study) the score of `rape*` may be offset accordingly. The scores of the matched patterns will contribute to the overall score of the parent category.

Assuming that `rape` has a score of 1.5, and `molestation` 2 and `sexual assault` has no score. `Molestation` occurs 10 times, `rape` 3 times, `sexual assault` 1 time. If `sexual assault` has not been assigned a score then the dictionary report will assign the score of 1 (1 x 1) and `rape` 4.5 (3 x 1.5) and `molestation` 20 (10 x 2). If, in addition, `molestation` is assigned the score -1, the report will assign `sexual assault` 1 and `rape` 4.5 as before, and `molestation` will be assigned -10 (10 x -1). This method of scoring allows to distinguish between an analysis where patterns are separately scored, and one where all the patterns in a category are treated as exchangeable.

As mentioned before, `Yoshikoder` allows for testing of differences between documents using the *risk ratio*.

$$RR = \frac{P_{Exposed}}{P_{NonExposed}}$$

For example, if two documents A and B are compared using a dictionary containing a category positive that represents positive language, then the *relative risk* for A and B is how much more (or less) likely it is to see a positive word in A than in B, expressed as a ratio. If A contains 100 words, 10 of which are positive and B contains 200 words, 15 of which are positive, then the risk ratio for the category positive is 10/100 = 0.1 (the probability of seeing a positive word in A), divided by 15/200 = 0.075 (the probability of seeing a positive word in B). 0.1/0.075 is approximately 1.333, implying that it is about 33 percent more likely to see a positive word in A than in B.

$$RR = \frac{10/100}{15/200}$$

`Yoshikoder` also computes a .95 confidence interval around this estimate. The interval indicates whether the ratio is significantly different from a ratio of 1 (1 in this case meaning that there is no true difference in proportions), by applying:

$$CI = \log(RR) \pm SE * z_{\alpha}$$

²⁵ In the tested version (0.6.3-preview3) the multi token pattern matching does not work. At least not for patterns like 'Peop* of Burma' or 'peopl* Burma' or simple 'people of'.

The standard score which Yoshikoder is using is 1.96, therefore it computes the interval on the 0.95 significance level. When a category does not appear in one of the documents the relative risk is reported as plus or minus infinity and no confidence interval is provided. If the category does not occur in either document then no estimate is provided. If the category relative frequencies are outside the bounds of the interval, the category count is marked as statistically significant difference.

IV.3.3 Concluding remarks

Yoshikoder is a simple, but a very intuitive application. It does not contain sophisticated parsing rules as the software mentioned so far and neither any NLP processing routines, but it has the infrastructure that allows to plug in different processing resources - for example, plugins for segmenting text for some non-latin languages, notably Chinese. It contains a graphical user interface and does not suffer from some limitations that seem very odd in the current circumstances – for example TABARI can process only ASCII text, lines must have certain size, otherwise be truncated – Yoshikoder on the other hand can work with texts written in any encoding and both in the server as well as workstation mode.

Author of the program clearly felt a need to develop such a tool, and as he says, it is also a way to promote content analysis standards. Lowe also maintains dictionary resources converted into the xml format suitable for import into the Yoshikoder.²⁶ As an example, it shows that even a rather simple processing software is still useful to the CA community and that the certain commercial software packages may be overpriced.

IV.4 SEMAN

SEMAN was created to test the possibilities of the USL, but it is also a tool for content analysis so we can describe it as such. However, the idea and also the name SEMAN comes from the previous work of Vladimir Smetacek. There existed a version of the program which was built and used in 80s for automatic keywording²⁷ of abstracts on the metallurgic databases in the former centre of information industry of Czechoslovak republic. This version was programmed for the mainframe computers and is not available any more, being long forgotten and non-functional. Vladimir Smetacek attempted to revive the idea of SEMAN several times in the past years, but it was only this reincarnation of the system that contains the components of a fully functional system.

²⁶ See: <http://www.yoshikoder.org/resources.html>

²⁷ But as an interesting coincidence, we can compare functionality of SEMAN against a similar tool which is used at the Centre for High-Energy Physics (CERN) for similar purposes. Based on the dictionary, it extracts (recommends) keywords that should be used for the bibliographic records.

SEMAN is written in Python, but certain heavy-duty components are built using Java and C++. However, all the components are fully controlled from Python²⁸ and can be executed on any modern operating system. It is run routinely on Windows, Linux and MacOS X. SEMAN supports unicode and is therefore able to process texts written in any encoding. It can run both in a graphical user mode, and also on the server in the console mode and was released as open-source. The motivation is similar to that of Will Lowe, the creator the Yoshikoder – to make the procedure of content analysis more transparent and also the tools affordable. The following sections display the internals of the system, and in the following chapter we will evaluate SEMAN performance and capabilities on several corpora.

For the purposes of description, we will follow a default pipeline of processing set of documents that consists of 3 steps:

1. NLP pre-processing
2. Translation into the semantic codes
3. Analysis and export of the results

IV.4.1 NLP pre-processing

Certain tasks are better handled by components that are specialised for it, SEMAN was therefore designed to take maximum advantage of the existing research and to separate the processing into disparate layers. The delegation of tasks onto other libraries is especially the case of the NLP processing operations, the topic of the following section.

In the current state, SEMAN is using two NLP frameworks – GATE and NLTK.²⁹ GATE (Cunningham, Maynard, Bontcheva, and Tablan 2002) is architecture, framework and development environment for LE (Language Engineering) written in Java. It is a very robust platform actively developed since the mid-1990s and used in many projects, especially for text engineering.³⁰ The GATE framework comprises a core library and a set of reusable LE modules, ranging from the document format conversion, POS tagging, sentence parsing to machine learning algorithms, neural networks or visual annotation tools.

28 Communication between the Java and Python components is made possible using the standard Java Native Interface (JNI). JCC – an automated code generator – produces a C++ object interface wrapping of Java libraries via Java Native Interface (JNI) that conform to Python C type system, which means that Java processes are directly available to the Python interpreter. When generating Python wrappers, JCC produces a complete Python extension module. SEMAN thus retains the ease of use of the Python language and its fast prototyping and friendliness of the dynamically compiled scripting language, but JCC allows us to offer very powerful and complex processing of the native NLP and information retrieval toolkits without sacrificing quality.

29 Though it is possible to include another NLP framework – UIMA, indirectly through GATE.

30 GATE version 1 was written in the mid-1990s; in 2000 completely rewritten in Java; at the time of this writing in version 6.5.

The framework implements the architecture and provides (amongst other things) facilities for processing and visualising resources, including representation, import and export of data. The components can be freely combined into processing workflows. SEMAN is able to execute any non-visual GATE workflow and to my knowledge, it is the only system that allows this kind of operation from inside Python. This flexibility is used by SEMAN itself as any type of non-visual³¹ workflow can be prepared in GATE and executed by SEMAN.

Of a special importance is a component called ANNIE (A Nearly-New Information Extraction System) which was for the information extraction tasks. Information extraction is a subfield of computational linguistics concerned with a constrained form of natural language understanding - only prespecified information is acquired from textual data.³² In the extraction, the machine attempts to discover what is relevant inside the document. (Cunningham 2005; King and Lowe 2003)

ANNIE consists of the following main processing resources: tokeniser, sentence splitter, POS tagger, gazetteer, finite state transducer (based on GATE's built-in regular expressions over annotations language ...), orthomatcher and coreference resolver. The resources communicate via GATE's annotation API, which is a directed graph of arcs bearing arbitrary feature/value data, and nodes rooting this data into document content (in this case text). (Cunningham, Maynard, Bontcheva, and Tablan 2002)

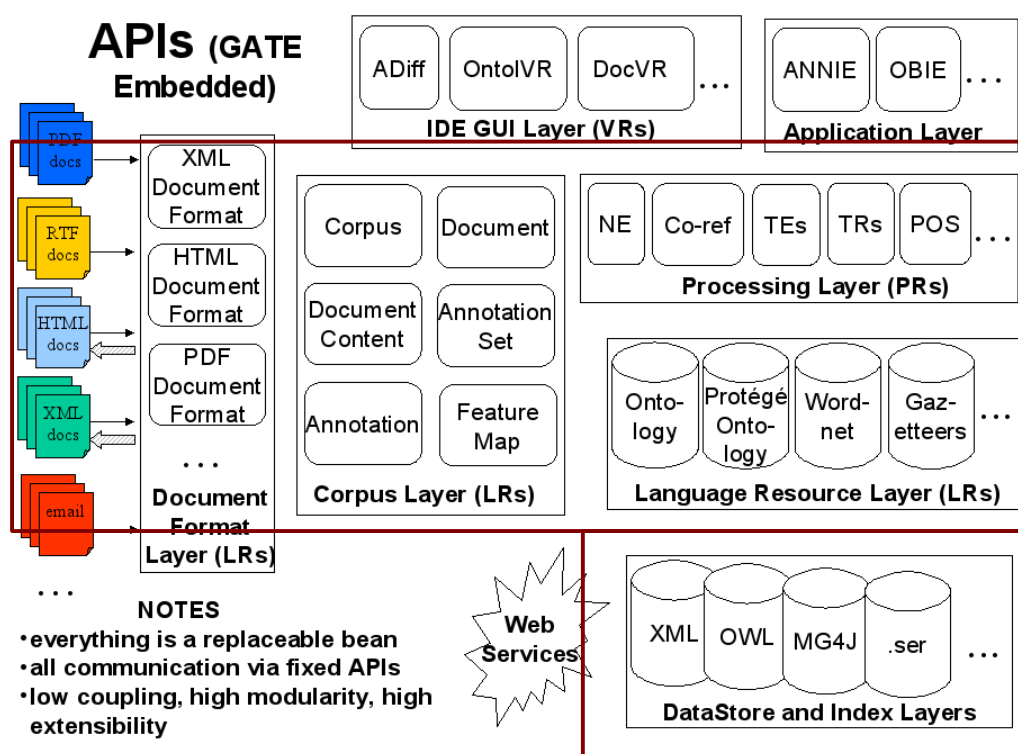


Illustration 16: General overview of GATE, the red area indicates the area of components which can be currently used by SEMAN.

In the default workflow, the usual processing of a document follows these steps:

³¹ I.e. workflow that does not use GATE's display components.

³² For a review of IE field, see (Turmo, Ageno, and Català 2006)

1. Document is converted from its original format (pdf, doc, html...) into a xml representation (GATE, as shown in Fig. 16 above)
2. Resulting GATE document is processed by ANNIE (based on the configuration, the default workflow will be described below)
3. Results are exported as XML with inline annotations
4. SEMAN reads in the xml produced by GATE and creates a collection of tokens
5. This collection is translated by SEMAN (depending on the currently configured workflow – will be described later)
6. Results are then summarized and exported/saved

The following chart illustrates the first pre-processing stage. GATE converts a document from its original format into an internal data structure.

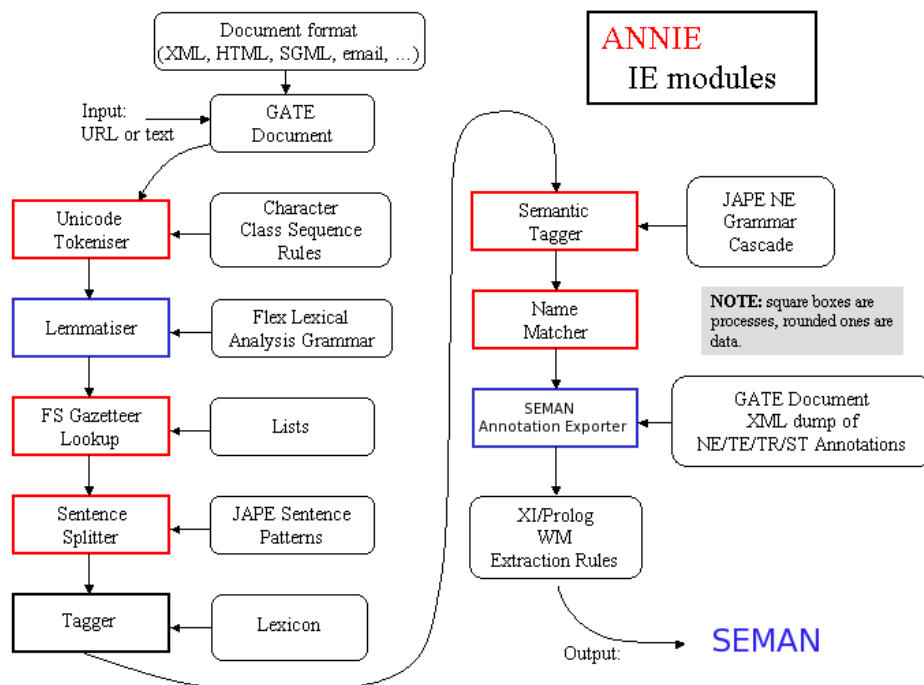


Illustration 17: Components of the ANNIE subsystem and the flow of a document.

The **tokeniser** splits text into simple tokens, such as numbers, punctuation, symbols, and words of different types (e.g. with an initial capital, all upper case, etc.). The aim is to limit the work of the tokeniser to maximise efficiency, and enable greater flexibility by placing the burden of analysis on the grammars. This means that the tokeniser does not need to be modified for different applications or text types.

The **sentence splitter** is a cascade of finite state transducers which segments the text into sentences. This module is required for the tagger. Both the splitter and tagger are domain and application independent.

The **tagger** produces a part-of-speech tag as an annotation on each word or symbol. Neither the splitter nor the tagger are a mandatory part of the NE system, but the annotations they produce can be used by the grammar (described below), in order to increase its power and coverage.

The **gazetteer** consists of lists such as cities, organisations, days of the week, etc and it enriches the tokens. It not only consists of entities, but also of names of useful indicators, such as typical company designators (e.g. 'Ltd.'), titles, etc. The gazetteer lists are compiled into finite state machines, which can match text tokens.

The **semantic tagger** consists of handcrafted rules written in the JAPE (Java Annotations Pattern Engine) language which describe patterns to match and annotations to be created as a result. JAPE is a version of CPSL (Common Pattern Specification Language) which provides finite state transduction over annotations based on regular expressions. A JAPE grammar consists of a set of phases, each of which consists of a set of pattern/action rules, and which run sequentially. Patterns can be specified by describing a specific text string, or annotations previously created by modules such as the tokeniser, gazetteer, or document format analysis. Rule prioritisation (if activated) prevents multiple assignment of annotations to the same text string.³³

An example of such a rule for matching \$20, \$20 USD, CZK 30.5 is:

Rule:

```
MoneyCurrencyUnit
(
  (AMOUNT_NUMBER)
  ({Lookup.majorType == currency_unit})
)
:number -->
:number.Money = {kind = "number", rule = "MoneyCurrencyUnit"}
```

Rule:

```
MoneySymbolUnit
(
  ({Token.symbolkind == currency}|
  {Lookup.majorType == currency_unit})
  (AMOUNT_NUMBER)
  (
    {Lookup.majorType == currency_unit}
  )?
)
:number
-->
:number.Money = {kind = "number", rule = "MoneySymbolUnit"}
```

The **orthomatcher** is another optional module for the IE system. Its primary objective is to perform co-reference, or entity tracking, by recognising relations between entities. It also has a secondary role in improving named entity recognition by assigning annotations to previously unclassified names, based on relations with existing entities.

The GATE document is finally converted to XML with inline annotations and passed to SEMAN. All tokens and annotations (or their subsets) are converted into the Python data structure and made available for further processing that will be described later.

33 Detailed documentation of the rules is available at: <http://gate.ac.uk/sale/tao/splitch8.html>

As was said earlier, the user has a lot of flexibility in the configuration of the previous steps. For SEMAN, this pre-processing stage is completely optional, it is also possible to use Python Natural Language Processing Toolkit, or ignore any advanced NLP processing altogether. If there is no dedicated layer, SEMAN input capabilities will be roughly equivalent to the capabilities of the already mentioned content-analysis tools. With the dedicated layer, SEMAN is superior in many aspects. Such superiority is however paid with the number of resources needed, and we will explore different scenarios in the following chapters – using dedicated NLP processing, only certain parts, different NLP engine or no NLP processing at all.

IV.4.2 Translation into semantic codes

The core task of SEMAN is to find indicators in the text, translate them into the correct USL codes, store such codes and analyse and export them for further processing. This chapter will describe the matching algorithm, the following sections will explain more about dictionary maintenance and document storage facilities.

As discussed previously, there are three main components of the dictionary entries:

1. prefixes
2. radixes
3. suffixes

The algorithm used for matching tokens is a simplified version of a morphological analysis. Simplified, because we can use the results of the NLP pre-processing and for the cases where the languages exhibit more complex problems than this algorithm can handle, we can first parse lexemes using the appropriate morphological tagger for the given language (if available, either in GATE, UIMA or NLTK), and obtain the basic lemmas together with POS information.

When identifying the pattern, SEMAN first constructs the possible prefix-radix-suffix combinations for the given token, and from these candidates selects the one which is considered best – the default functionality simply selects the longest matching radix. The matching with longest radix can be made sufficiently unambiguous if the dictionary is constructed properly, as results of the experiments show, see V.1 The pattern matching mechanism, p. 115. But in most cases the default behaviour is sufficient and it is easy to rectify parsing exceptions by changing the dictionary, for example by listing the full form of words, or selecting the correct translation based on the contextual information (for example using appropriate POS tag).

Translation of the individual token uses the following algorithm:

```
# try to match directly
direct_lookup_indexes = get_indexes(language)
for index in direct_lookup_indexes:
    if index.has(token_value)
        and callback_check_lookup(token, index.get(token_value)):
            update_token(token, features to save...)
            return True
```

```

# no matches found using lookup methods of the indexes implementations
prefixes = self.getPrefixes(language)
suffixes = self.getSuffixes(language)

# try to find prefix-radix-combinations
suffix_candidates = find_matching_suffixes(token_value, suffixes)
prefix_candidates = find_matching_prefixes(token_value, suffixes)

if len(suffix_candidates) > 0 or len(prefix_candidates) > 0:
    possible_matches = []

    radices = self.getRadices(language)

    if len(suffix_candidates):
        #we start with suffixes
        for (suffix_candidate, suffix_entry) in suffix_candidates:
            slength = len(suffix_candidate)
            word = token_value[0:-slength] #strip-off suffix
            for (prefix_candidate, prefix_entry) in prefix_candidates:
                plength = len(prefix_candidate)
                root = word[plength:] #strip-off prefix
                #and see if the resulting quasi-lemma is in our dictionary
                if root in radices:
                    possible_matches.append(prefix_candidate,
                                            root,
                                            suffix_candidate,
                                            sem=[prefix_entry, radices[root], suffix_entry] )
            else: # if we do not have prefix
                #we search in dict of radices
                if word in radices:
                    possible_matches.append(
                        prefix_candidate,
                        root,
                        suffix_candidate,
                        sem=[None, radices[root], suffix_entry] )
        else: # no suffixes
            for (prefix_candidate, prefix_entry) in prefix_candidates:
                plength = len(prefix_candidate)
                w = token_value[plength:] #strip-off prefix
                if w in radices:
                    possible_matches.append(
                        prefix_candidate,
                        root,
                        suffix_candidate,
                        sem=[prefix_entry, radices[root], None] )

    if len(possible_matches) == 0:
        record_failure(token)
    else:
        features_to_save = select_the_best_match(token, possible_matches)
        update_token(token, features_to_save)

```

As an example, given the following dictionary entries:

```

esq revol = abc dde
esq french revol = abc dde fra

```

and this set of suffixes and prefixes:

```

ess ution = 0
ess utionary = 0
esp pre = 0
esp contra = xui

```

The following patterns will be recognized (codes assigned):

```

revolution = abc dde 0
revolutionary = abc dde 0

```

```
prerevolutionary = 0 abc dde
contrarevolution = xui abc dde
```

However, because the dictionary is very simple, these tokens will not be recognised:
 revolutions, contra-revolution, contra-revolutions,...

In order to match these tokens, the dictionary must be enhanced by additional suffixes – or list the full patterns in the dictionary. But listing the full form of the token(s) is cumbersome and recommended only for exception handling. For example we would like to count tokens *pressure*, that stand at the position of a noun, but we do not want to use the POS tokenizer because it slows down the processing or because we are processing a corpus of documents from a very limited domain where the ambiguity is limited. If for the purpose of our application, these requirements are sufficient, we can use the following entries:

```
hsq pressur = code1 code2 code3
hsq pressures = XXX #entry will be ignored
```

Application of this rule on the corpus of HEP papers (full details available in the next chapter) yield correct results, the first entry matches the following word forms: *pressure* (1049), *Pressure* (18), *pressurized* (3), *pressures* (61), *PRESSURE* (2) – while the second entry filters out the considerable number of matches of the form: *Pressures* (1), *PRESSURES* (2), *pressures* (61). As to the rest, such a simple mechanism of negative exceptions is used for TABARI and authors report that it is responsible for dealing with the most frequent cases of ambiguity, roughly 75% of the ambiguous cases (Schrodt 2009, 7-8) However in the case of SEMAN this strategy would not be as successful if we work with individual words only.

SEMAN is of course able to recognize not only individual tokens, but also groups of tokens, provided they follow in sequence or dedicated post-processing is activated. For example, this dictionary entry:

```
hsq cosmol model = 00mrl
hss ogy = 000
hss ogical = 000
hss s = 000
```

Will indentify:

```
cosmological model, cosmological models, cosmology model
```

But will not match:

```
models of cosmology, cosmologic model, cosmological modelling
```

Yet because SEMAN identifies the basic elements of these patterns, it is also possible to match the groups of tokens (lookbehind for matches that appeared before the entry, lookahead for matches after the entry, or both-directional lookup) - up to certain distance or until a boundary is reached. Such an extension is a part of the SEMAN standard toolkit. SEMAN can find the groups of tokens, in or out of order. Taking for illustration again the High-Energy physics thesaurus, we can find following entries there:

```
charmonium: mass: calculated
```

```
charmonium: decay constant: calculated
```

These are composite keywords that signal occurrence of a certain concept. In order to identify their presence (and we could do it inside one sentence, or a paragraph) each of the components needs to be listed as a single pattern, I.e.

```
hsq deca constant = ...
hsI charmonium = ...
hsI mass = ...
hsq charmonium: deca constant = ...
hsI charmonium: decay constant: calculated = ...
hsI charmonium: decay constant: measured = ...
hsI charmonium: decay modes = ...
```

Given this definition, and if we set the distance of the tokens to $w=5$, the following sentences would produce the match:

```
The charmonium decay constant was calculated...
Calculation of the decay constant of charmonium was...
```

Because internally, SEMAN does not work with words, but USL codes, the group of words that read `decay constant` is identical to group of words with a different reading, such as `decayed constant(!)`, `decay constants`, or `deca constants(!)`. Of course, if the dictionary is designed carefully this feature can be used for the benefit of better matching. Because semantic codes act as **unique identifiers**, we can change freely the wording (the left-hand side part of the entry). While the pattern matching words will change, the combinations remain the same. In fact, the semantic codes are for SEMAN more important than the reading of the words – this reverses the standard view of the most matching schemes, where people think of the concepts in terms of words. In SEMAN we think of concepts in terms of concepts. If we added a new definition:

```
hsq constant: deca = ...
```

Provided the right side of the entry remains the same, the following two entries will be for SEMAN identical: `decay constant`, `decay is constant`. Nothing needs to be changed also in the definition of the group of composite tokens and following sentence would identify `charmonium: decay constant: calculated`:

```
The constant of the decay of charmonium was calculated using...
```

But SEMAN matches elements in or out of order and it is only possible to limit the distance of tokens between each match and the mechanism has its limitations, consider the following:

```
Charmonium and charmed meson decay constants were used in
calculation of..
```

This sentence would produce the same match, but semantically the `charmonium` constant is not calculated, something else was calculated. Therefore this is clearly not what we wanted to extract. The simplistic matching of tokens is clearly not capable of distinguishing such situations.

For the purposes of content analysis, this was not considered as a main problem as we were not interested in interpreting individual sentences correctly. Instead signals are sought in a (possibly very large) corpora, therefore certain amount of errors is expected and tolerated. However it would be possible to solve also this situation – and in several ways: we could change the pattern in the dictionary to further limit the forms of the matched entries, or we can use the part-of-speech tagger and ignore matches that are composed of transitive passive verbs, but probably the most accurate approach would be to fully parse the sentence and identify individual clauses and function of each word. With a high accuracy, we could remove the false positive matches.

While SEMAN does not use the sentence parsing by default, it certainly has this capability, however such processing would increase the time needed for analysis so it was not used in the experiments. But we did not mention another available mechanism of code callbacks. They provide an interesting alternative of how to increase the accuracy of the parsing mechanism maintaining the high-speed processing.

SEMAN internally works with the stream of tokens (to the engine they constitute acyclic graph³⁴) and the token object usually corresponds to a word, but it can contain anything from a compound word to a sentence. It holds the features, which can be set and retrieved. The specificity of the token object is that every token has access to all the other tokens in the collection, as they appeared in the text. So that the context can be explored. It is possible to write special routines (in Python) and register them as callbacks inside SEMAN. For example, for each translation, we can test whether the tokens around the translated token have appropriate POS tag. These callbacks normally receive only the currently processed token, but because the token has access to all of its neighbours – the callback can freely change the whole collection.

In the current version of the SEMAN translation, the following callbacks can be registered:

1. Callbacks that prepare tokens (pre-processing, work mostly with the lexemes):
 - **tokenize_input**: this is a general tokenizer point, it receives a text and should return a collection of tokens that will be processed
 - **check_token**: checks individual token and decides if it should be processed or ignored
 - **prepare_token**: cleans up the token and prepares it for translation
 - **find_next_part**: when identifying multi-word tokens, this callback receives the currently processed token (already translated and only if it was translated), and should return the correct next key from the dictionary of multiwords. By applying

³⁴ If we were processing arabic languages or Jewish, the processing is no different than latin or other languages with left-to-right scripts. The only difference would be in the construction of the dictionary. Because the simplified-morphological matching is symmetrical, for right-to-left languages we would have to put prefixes at the place of suffixes and vice versa.

special function, we can effectively match words that are separated by other tokens. For example the series of tokens: **Primakoff** found the **effect...** can be identified as containing one token: **Primakoff effect**

- **token_merged**: callback called when SEMAN identifies a multi-word token, for example originally there were two tokens: 1.) `constant`, 2.) `decay`, but after this call, these tokens were removed from the collection and replaced with one token: `constant decay`

2. Callbacks that are concerned with the process of parsing the lexemes and their translation:

- **token_transl_success**: called when translation is finished and token was found in the dictionary, this callback can approve or disapprove of the translation, it also returns attributes to be cached, or nothing to prevent its caching
- **token_transl_failure**: called when NO translation was found for the given token, this callback can activate a special processing, change the token values and call the translation again. If something is returned, that value will be cached as in the success callback - and success will be reported
- **translation_problem**: general method called when the token does not have one code, but several codes and translation cannot decide which semantic code should be chosen
- **morphoanalysis_ambiguity**: we have identified several possible combinations of prefix-radix-suffix that correspond to the current token. And we were unable to decide which of them should be selected. This method can go through the list of candidates and select the right one
- **emit_multiple_definitions**: used usually in debugging stages, called when the translation identified several unique codes, but could not decide which of them to choose
- **disambiguate_multiple_definitions** - this is the same case as above, with the difference that the callback should choose one definition.
- **veto_direct_lookup**: the exact match of the token was found in the dictionary, the normal behaviour is to stop any translation attempts and continue with the translation of the next token. This callback can veto such an operation and order SEMAN to continue with trying to find other translations. This callback can also select the desired translation from the list of possible translations, if there are multiple matching entries in the dictionary for this token.
- **longest_radix_ambiguity**: when the parsing algorithm finds several possible parses of the word, all of them with the corresponding entry in the dictionary, this callback can decide which one should be used.
- **before_translation**: callback activated before the whole collection of tokens is translated, it can be used for filtering the tokens or any similar operation
- **after_translation**: callback activated after the whole collection of tokens was translated, this is the callback where composite matches are identified or when the translation is checked and spurious cases resolved

3. Operational callbacks that influence the results of the translation:

- **prepare_dict_value**: this callback is used when we need a special attributes of the dictionary, for example the POS information (if that is necessary in further steps) or it can automatically create a stemmed version of the dictionary entries and similar. What this method returns will be used as a translation result (usually it is a string - eg. 'code1 code2 code3', but it can be also a complex object with attributes – usually used in cases when we want to disambiguate words and look at the context of the previously translated tokens)
- **token_get_keys**: returns a value of the token that will be processed and also a key identifying this token in the cache of other tokens – if found, then the translation is not repeated. Cachekey is key under which results is found/saved into cache for faster lookups. If None is returned, cache is effectively deactivated

The callbacks can be registered and re-registered freely when needed. Clearly, this option is reserved for the programmable usage of SEMAN, but it offers a interesting alternative to the more powerful, but slower NLP processing. This is also the main point of the mechanism. It can provide for fast-enough and robust solution of many usual cases, but in situations when we need to process more complex patters, it allows us to use a very specific processing logic. The level of complexity depends on the data that are available to us and also on the purpose of the application. If we face the difficult texts, we can first pre-process the input text with a more powerful NLP tools and then ask SEMAN to do the translation. The modular design allows for a wide variety of choices.

IV.4.3 Storage and analysis

Out of the three tools already mentioned, only TABARI keeps the record of the already processed results (in plain text format). The other two work with documents without storing results of the previous phases. SEMAN can work in such a way also, but it also provides a mechanism for incremental processing, actually two mechanisms:

1. RDBMS storage
2. Lucene index storage

The RDBMS storage keeps documents translated into USL codes, the tokens are saved into the database in the following format

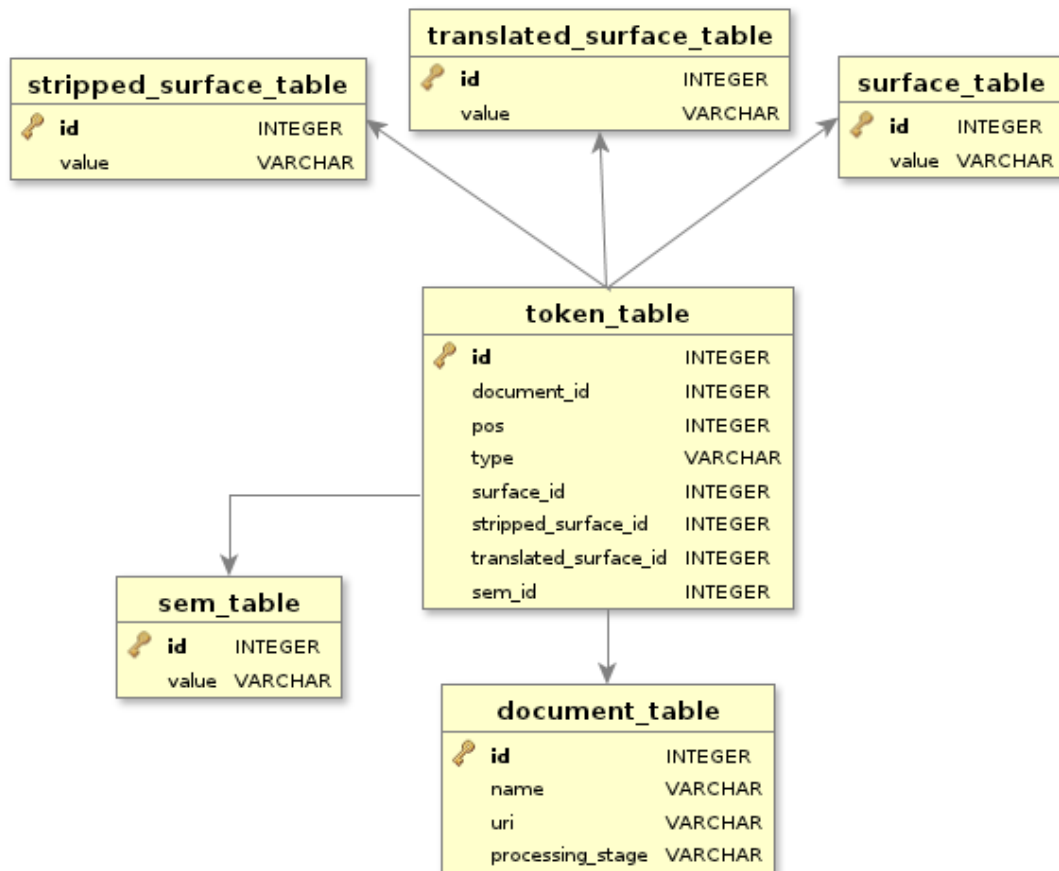


Illustration 18: Database schema of the document storage; built around tokens and their translation.

Internally, the SQLAlchemy (“SQLAlchemy - The Database Toolkit for Python” n.d.) toolkit is responsible for the database layer management, the data and the model are decoupled so that SEMAN is free to disregard differences in the implementation of the underlying database engine. The data abstraction layer also allows for construction and manipulation of SQL expressions in a platform agnostic way, and offers easy to use and fast result objects, as well as table creation and schema reflection utilities. If there was a need to implement a storage not in the RDBMS system, but on a file system or a distributed filesystem, nothing would change inside SEMAN. The changes would happen at the level of the database abstraction layer.

The performance depends on the underlying RDBMS engine. In the basic settings, SEMAN is using the sqlite implementation, which is both portable and relatively fast, but it does not scale well for big collection of documents. In such cases, more powerful solutions such as MySQL, Postgresql, or Oracle should be used. However, such a requirement would complicate the life of most users, so a second search-engine index-based mechanism was

implemented. It is built on top of the Lucene information retrieval library³⁵ – this implementation can handle relatively very large volumes of data and provides a special mode of analysis not available with the RDBMS implementation.

After the documents are translated, number of features are stored in separate field (indexes) – later on, they can be retrieved and exported for analysis. This mechanism makes it possible to store any feature associated with the word/token, however a special treatment is given to the USL semantic codes. They are stored and indexed together with the original text at the same position as the original text. The situation is similar to adding new dimensions to words, for example the sentence:

The ATLAS detector was stopped.

Will be first parsed by GATE, then by SEMAN, the text split into tokens, then analysed and translated. ATLAS for example is recognized as a known concept (a composite of: instrument + experiment + particles + high energy physics + CERN experiments). In the language of semes denoted as:

0005s 0009d 450is 23ioi 8sdet

The complete sentence then has a form similar to:

1. The
2. ATLAS detector, 0005s, 0009d, 450is, 23ioi, 8sdet
3. was
4. stopped

We process this structure in a special way inserting **all the semantic codes** at the same position as the original text. Lucene will build the index where several tokens are at the same position:

token	position
-----	-----
ATLAS	0
detector	1
0005s	0
0009d	0
450is	0
23ioi	0
8sdet	0
was	2
stopped	3

The first position 0 holds 6 tokens, namely: particle, 0005s, 0009d, 450is, 23ioi, 8sdet. The word detector is placed at position 1, because of the limitations of the *PhraseQuery* of the Lucene engine – if we wanted to search for phrases, the words need to be adjacent, and not at the same position. So the phrase query “ATLAS detector” matches only if the tokens' positions are different. The other parts of the sentence were not recognized by SEMAN, and therefore not enriched with the semantic features.

35 <http://lucene.apache.org/java/docs/index.html> [Accessed: 24-7-2011]

While the features can be stored independently in separate indexes, the method described above allows us to search for surface word forms and at the same time look and work with relations between semantic codes, ignoring the natural form altogether. Semantic features influence search results (sort order) in many ways:

- recognized concepts can be boosted during indexing (ie. some words become more important)
- we can translate query and boost recognized concepts during search
- we can search for synonymous concepts (i.e. "0005s AND 0009d AND 450is AND 23ioi AND 8sdet" at position X)
- we can search for similar concepts (i.e. any of '0005s 0009d 450is 23ioi 8sdet' IF they occur together up to a certain distance)³⁶

This storage mechanism is then utilized during the analysis of statistical significance when searching for combinations of *semes*.

IV.4.4 Search for statistically significant cooccurrences

This section will provide an overview of the theory behind our search for semantically related components. How we use the methodology of the lexical semantics for the application on the semantic features, what statistical tests and tools are used in the search for semantically related and statistically significant combinations of *semes*.

English linguist J.R. Firth acquired fame for his approach to linguistic analysis based on the view that language patterns cannot be accounted for in terms of a single system of analytic principles and categories, but that language is a system that can be described by a group of different subsystems at different places within a given level of description. One of Firth's phrases, describing the search for nature of words, is particularly known nowadays : "You shall know the word by the company it keeps!" (Firth 1957, 11) And if we replace 'word' with 'concept' in the very same phrase, we will look at a very similar topic but from a new perspective. For this to work, let me first describe what the lexical linguistics and natural language processing understand under the two key concepts of *cooccurrences* and *collocations*.

The term collocation was first used in 1930s to point at characteristic, "habitual" word combinations. As Evert (Evert 2005, 15) describes, different definitions of the term were introduced over the past fifty years, and lexicographers often disagreed about border cases. Nevertheless there was always a clear consensus that **words are not combined randomly**. And rules of grammar and syntax on their own can only explain part of the story, perhaps the smaller part of the whole story because these combinations are an important source of information not only about the language itself but also about the world we live in and what

³⁶ Note: the last two examples don't describe a Boolean query, we are not searching for: "0005s OR 0009d OR 450is OR 23ioi OR 8sdet" in the whole text, we will search only for groups of codes that MUST occur together up to certain chosen distance.

and **how** we think. The linguistically motivated search into the ways people combine concepts (and not words) was also one of the main motivations for Vladimir Smetacek to start work on SEMAN in the first place.

Over the past fifty years different methods of the study of collocates evolved, according to Evert we could distinguish two main branches to the study of collocations: a *distributional* and an *intensional* school. The distributional approach is the direct successor to Firth's ideas and evolved in the UK. This approach is often referred to as Neo-Firthian and is based on the premise that collocations are **recurrent** word combinations in a particular text, that these combinations are directly observable and no linguistic pre-processing (or very little pre-processing) is needed for their identification. Taken to the extreme, the corpus is the only and definitive authority in what can be found and no linguistically motivated pre-processing should tamper with the data. The most notable figure of this school is the british linguist M.A.K. Halliday, for an overview of the school itself see (Williams 2003)

On the other hand, the intensionally motivated search for collocations is different. Collocations are not viewed as purely recurrent word combinations, as if they were automatically extracted from the source documents, they also have to belong to certain linguistically motivated types or classes of words. Linguists will define rules of what is considered to be a valid collocations. This approach is naturally more inclined towards dispute and argumentations over the linguistic theories, but usually collocations are placed somewhere in the grey area **between fixed idioms** ('kick the bucket') and **free combinations** ('outstanding achievement'). In a narrower sense, they are understood as semi-compositional word pairs, with one "free" element (the *base*) and the other element that is lexically determined (the *collocate*). While the free element retains its independent meaning in the combination, the collocate often contributes a meaning component that the collocation cannot have on its own. And the resulting combination of the elements creates a **new concept**, something that is more than a mere sum of its parts.

The intensional approach is then characterized by varying and much more elaborate requirements on conditions of collocations; more extra textual linguistic knowledge is often required for the analysis. If the search is done in an automated manner, parts of speech of the individual words, as well as their function inside the sentence are identified, and obviously words are parsed and interpreted using whatever linguistically motivated theory was behind the tools – and this is also an important argument in favour as well as against the intensional approach. The pre-processing transforms the source data and researchers might ask questions that are not stemming from data, but from their own beliefs of the language.³⁷ For an overview of the theoretical approaches of this school, see (Bartsch 2004, 27-64).

Evert, which is an excellent source for evaluation of the statistical measures used for the collocation search, summarizes his understanding of the differences between the schools in the following way:

In order to make a clear distinction between the two approaches to collocations, I refer to the distributional notion as **cooccurrences**, which encompasses both the observable (cooccurrence) frequency information and its interpretation as an indicator of statistical association. This description seems fully adequate for the Neo-

³⁷ A striking parallel with the same problem of content analysis, as described in III.2.1.1 Reliability and validity

Firthian understanding of a collocation as a recurrent word combination, cf. the definition “collocation is the occurrence of two or more words within a short space of each other in a text” (Sinclair 1991, 170). By contrast, I reserve the term *collocation* for an intensionally defined concept that does not depend on corpus frequency information.” (Evert 2005, 17)

He further continues to propose the Manning and Schütze's definition of the collocation as “word combination whose semantic and/or syntactic properties cannot be fully predicted from those of its components, and which therefore has to be listed in a lexicon”, collocations “correspond to some conventional way of saying things” (Christopher D. Manning and Schuetze 1999, 151). As for the cooccurrences, Evert defines them as “observable evidence that can be distilled from a corpus by fully automatic means. After statistical generalisation, this information can be used to predict which word combinations are most likely to appear in another corpus.” (Evert 2005, 23)

Collocation is therefore a generic term that attains a very specific meaning based on the requirements of a particular research question or application. The distinction between the two different approaches is then highly relevant for us and the study of semantic features because we can draw the direct parallel between collocations and the combination of semantic features. And also the two, somewhat distinct, ways of processing will be applied to them later – I can refer to the purely frequency based combinations of the semantic features as **sem cooccurrences**, and reoccurrence of specific categories of semantic features as **sem collocates** – for the latter, we need to introduce and operate with extratextual information that cannot be found or seen in the corpus and the token positions itself. Perhaps we will be required to find these criterias ourselves.

IV.4.4.1 Types of cooccurrences

In the search for sem cooccurrences and collocates we will need to use statistical tests of significance. The corpus is used as a source of information about elements and their combinations, and thanks to random sampling and the law of big numbers, we try to establish whether the combinations that we discovered are due to chance and can therefore be disregarded, or whether there is indeed some important link between the elements and we have discovered sem cooccurrence or sem collocation. For any of the two approaches, we will work with frequencies of the elements and their pairs.

However the cooccurrence of semes can be defined at different levels: as a proximity, looking at the semes and their immediate neighbours – ie. searching a window of certain size. Or as occurrences of semes within the same linguistic or structural unit (sentence, paragraph, chapter, article etc) or as a functionally specific (usually syntactic) relationship between words and their semes – in this case, we limit our search to the semantic features that belong to a certain POS classes, and only certain combinations of them are allowed. Such an approach amounts to a filtering of the semantic pairs before the measurement.

Historically the positional search is older – words (in our case semes) are considered to form a pair if they appear within a certain distance of each other, this distance is usually measured by the number of intervening words, in the case of semes, by the number of intervening words or semantic compounds. In the literat-

ure, such a group is often referred as *collocation span*, so we could speak about *semantic collocation span*. The advantage of this approach is that data is directly observable and no special pre-processing is necessary to discover pairs.

However, this approach is not without problems. If we consider only the combinations of semantic tokens based purely on their distance, we are potentially creating too many pairs, but there is a more serious problem than this. It can be illustrated by an example. It was shown by Justeson, that extraction accuracy can be considerably improved by using a simple POS filter which accepted only certain types of relations, such as AJ+N, V+N etc. (Justeson and Katz 1995) Combining the frequency information with a small amount of linguistic knowledge the algorithm achieved 80% accuracy, which is a considerable increase. There is a number of other studies that confirm the same view that linguistic information when taken into account during the filtering of collocation candidates, considerably improves the accuracy and removes false negatives. (Evert 2005; Smadja 1993)

Such findings show that inside the corpus there exist many different kinds of relations and various types of relations follow substantially different frequency distributions. Statistical methods based on 'simple' frequency counts effectively mix these various distributions into one distribution, but arguably the obtained results could be considerably improved if the statistical tests were applied to a single "homogeneous" frequency distributions rather than to such a mixture. This is where the historically newer approach to the collocation search comes in place. It is centered around the idea that occurrences can be found using *linguistic interpretation* of corpus data. Therefore only certain distributions are generated and included in analysis. Typical examples contain dependency relations and subtrees in a phrase-structure analysis or relations based on the POS, and syntactic relations that are considered valid for English³⁸: (i) verb + noun (direct object), e.g. commit suicide; (ii) adjective + noun, e.g. reckless abandon; (iii) adverb + verb, e.g. tacitly agree; (iv) verb + predicative adjective, e.g. keep sth handy; (v) verb-particle constructions, e.g. bring sth up; and (vi) verb + prepositional phrase, e.g. set in motion. For knowledge extraction tasks, there is usually a strong emphasis on verb + noun relations. (Evert 2005, 19)

Such operation requires automatic linguistic pre-processing, and critics point out this makes the unbiased analysis of the observable facts impossible, because pre-conceived notions of a particular linguistic theory are inserted in the data. However, the advantages might overweight the risks, as the simple example above shows. While the processing is technically more complex, the final extracted data are actually more elegant and easily interpretable. And this usually means that we can get meaningful results. The explanation being, that in natural language words are not combined randomly into phrases and sentences so we cannot build a model of the language based on pure combinations³⁹ (for example certain related words may be very far in terms of position in a sentence). By focusing on the position, we are probably mixing the different distributions into one search space. And by mixing them, the Neo-Firthian approach is actually complicating the search. Whilst by

38 This type of analysis is dependent on the knowledge and interpretation of the language units, therefore it is not applicable universally, across languages.

39 Even if this approach of 'bag of tokens' is used successfully in many areas of NLP and yields significant results, therefore the advantages of the 'cleaner' distributions may not be always helpful.

separating the distributions, we can remove a lot of noise.

IV.4.4.2 Extracting cooccurrences from text

The following part explains how the extracted cooccurrences are evaluated and what methods are applied inside SEMAN. While many treatments of the subject can be found, for example (Dale, Moisl, and Somers 2000, Chpt 21; Christopher D. Manning and Schuetze 1999, Chpt 5) I have adapted the reference presented in the Evert's evaluation of the association metrics because he created a unified system and notation for it. Evert's main interest was the evaluation of association measures for the **relational data**, however the following section applies both to positional as well as relationally related searches.

First we have to gather data about pairs of tokens. Cooccurrence frequency data is analysed separately for each pair (u, v) . If we do the analysis in the relational paradigm, only certain types of tokens are considered – such as the aforementioned groups adj.+noun, verb + noun etc. In the positional paradigm, on the other hand, all adjacent tokens up to the length of window w is considered. The instances are considered to be valid pairs if their frequency is higher or equal to 1 ($f \geq 1$).

Evert shows that reliable statistical inference is impossible in principle for the hapax and dis legomena ($f = 1, 2$). “In this frequency range, quantisation effects and the characteristic highly skewed distribution of the cooccurrence probabilities of pair types (roughly following Zipf’s law) dominate over the random variation that statistical inference normally takes into account. As a result, probability estimates are entirely unreliable unless the precise shape of the population is known. This rather negative result provides theoretical support for the application of a frequency threshold, which should at least exclude the hapax and dis legomena ($f \geq 3$). Quantisation and the shape of the population no longer play a role for $f \geq 5$.” (Evert 2005, 166)

Usually a filter is applied that removes sparse pairs and thus only pairs with $f \geq 3$ are retained. Given a pair (u, v) , the extracted pair tokens are classified into the four cells of a **contingency table**, depending on whether the first component of the token belongs to type u and whether the second component belongs to type v . These conditions are written $U = u$ and $V = v$ in the contingency table shown below.

	$V = v$	$V \neq v$
$U = u$	O_{11}	O_{12}
$U \neq u$	O_{21}	O_{22}

$$O_{11} + O_{12} + O_{21} + O_{22} = N$$

The very same treatment is reserved for the positional data. In this table, u and v stand for the occurrences of the adjacent tokens in the corpus (up to the distance w), and the other cells mark the frequencies of the non-occurrence data. The cell counts O_{11} , ..., O_{22} of the contingency table are called the **observed frequencies** of the pair (u, v) . For example, the cell O_{11} contains the number of tokens that are of type AN (adjective-noun) from the corpus, O_{12} will hold the number of pairs where the type A is present, but the second component is made of every other type but N $freq(u, \neg u)$, similarly the cell O_{21} holds data about the pairs in which there N is present, but the second element is not of type A. Finally, the cell O_{22} holds the number of all the rest of the pairs from the corpus.

The cell frequencies add up to the total number of pair tokens, called the **sample size** N . As an example, the contingency table below shows the observed frequencies of the adjective-noun pair type (black, box) in the *British National Corpus*.

	$V = \text{box}$	$V \neq \text{box}$
$U = \text{black}$	123	13 168
$U \neq \text{black}$	1 810	4 951 883

The **row totals** R_1, R_2 and **column totals** C_1, C_2 often play an important role in the analysis of frequency data. They are referred to as **marginal frequencies** and are written in the margins of the table. R_1 is the marginal frequency of u , i.e. the number of pair tokens whose first component belongs to type u . Similarly, C_1 is the marginal frequency of v . O_{11} , the number of cooccurrences, is also called the **joint frequency** of the pair (u, v) .

	$V = v$	$V \neq v$	
$U = u$	O_{11}	O_{12}	$= R_1$
$U \neq u$	O_{21}	O_{22}	$= R_2$
	$= C_1$	$= C_2$	

Continuing with the example, we can see that the joint frequency of the pair is $f = O_{11} = 123$. The marginal frequencies are $f_1 = R_1 = 13,291$ (13,291 pair tokens of the form $(black, *)$) and $f_2 = C_1 = 1,933$ (1,933 pair tokens of the form $(*, box)$). The full sample consists of $N = 4,966,984$ adjective-noun pair tokens extracted from the *British National Corpus*.

	$V = \text{box}$	$V \neq \text{box}$	
$U = \text{black}$	123	13 168	$= 13 291$
$U \neq \text{black}$	1 810	4 951 883	$= 4 953 693$
	$= 1 933$	$= 4 965 051$	$N = 4 966 984$

The quantitative study of the frequencies requires a statistical model of the frequencies to draw conclusions about the observed values. The model assumes that the observed pair of tokens are drawn randomly from an **infinite population** (so that the much simpler equations for sampling *with replacement* can be used). Such an infinite population can be described as a set of (pair) types and their relative frequencies in the population, which are the **parameters** of the model. The random selection of a single pair token is modelled by two random variables $\mathbf{U}$ and $\mathbf{V}$ that represent the component *types* of the selected token. A **sample** of $\mathbf{N}$ pair tokens is modelled by $\mathbf{N}$ independent pairs of random variables $\mathbf{U}_m$ and $\mathbf{V}_m$, whose distributions are identical to those of the *prototypes* $\mathbf{U}$ and $\mathbf{V}$. Just as with the observed corpus data, the frequency information for a given pair type $(\mathbf{u}, \mathbf{v})$ is collected in a **contingency table** of random variables $\mathbf{X}_{11}, \dots, \mathbf{X}_{22}$:

	$V = v$	$V \neq v$	
$U = u$	X_{11}	X_{12}	$= X_{R1}$
$U \neq u$	X_{21}	X_{22}	$= X_{R2}$
	$= X_{C1}$	$= X_{C2}$	

X_{11} is the number of $\mathbf{m}$ for which $\mathbf{U}_m = \mathbf{u}$ and $\mathbf{V}_m = \mathbf{v}$ etc. The marginal frequencies are also random variables, written as X_{R1} , X_{R2} , X_{C1} , and X_{C2} . This contingency table contains all relevant information that a random sample can provide about the pair $(\mathbf{u}, \mathbf{v})$.

The sampling distribution is determined by the **probability parameters** $\tau_{11}, \dots, \tau_{22}$ of $(\mathbf{u}, \mathbf{v})$, which can be defined in terms of the prototype variables $\mathbf{U}$ and $\mathbf{V}$.

$$\begin{aligned} P(U = u \wedge V = v) &= \tau_{11} & P(U = u \wedge V \neq v) &= \tau_{12} \\ P(U \neq u \wedge V = v) &= \tau_{21} & P(U \neq u \wedge V \neq v) &= \tau_{22} \end{aligned}$$

Only three of the four probability parameters, which must add up to one ($\tau_{11} + \tau_{12} + \tau_{21} + \tau_{22} = 1$) are *free* parameters. Therefore, it is more common to use an equivalent set of three parameters, given by the equations below. π is the probability that a randomly selected pair token belongs to the pair type $(\mathbf{u}, \mathbf{v})$, π_1 is the probability that its first component type is $\mathbf{u}$, and π_2 that of $\mathbf{v}$ being the second component type.

$$\begin{aligned} \pi &= P(U = u \wedge V = v) = \tau_{11} \\ \pi_1 &= P(U = u) = \tau_{11} + \tau_{12} \\ \pi_2 &= P(V = v) = \tau_{11} + \tau_{21} \end{aligned}$$

The contingency table of random variables represents a **potential outcome** of the sample. The probability of a particular outcome, i.e. a contingency table with cell values $k_{11}, \dots, k_{22}$, is given by the **multinomial distribution** of $(\mathbf{u}, \mathbf{v})$:

$$P(\vec{X} = \vec{k} | N) = \frac{N!}{k_{11}! k_{12}! k_{21}! k_{22}!} \cdot (\tau_{11})^{k_{11}} \cdot (\tau_{12})^{k_{12}} \cdot (\tau_{21})^{k_{21}} \cdot (\tau_{22})^{k_{22}}$$

Each random variable X_{ij} has a binomial distribution by itself, so the probability of an outcome with $X_{ij} = k$ (regardless of the other cell values) is:

$$P(X_{ij}=k|N)=\binom{N}{k}\cdot(\tau_{ij})^k\cdot(1-\tau_{ij})^{N-k}$$

The **statistical association** between the components u and v of a pair type is a property of its probability **parameters** (i.e. a population property of the statistical model). Consequently, a main goal of the statistical analysis of cooccurrence data is to make **inferences** about the population parameters from the observed data. The tables below schematise this comparison between probability parameters and observed frequencies.

Population	$V = v$		$V \neq v$	
$U = u$	τ_{11}	+	τ_{12}	$= \pi_1$
$U \neq u$	τ_{21}	+	τ_{22}	$= 1 - \pi_1$
	$= \pi_2$		$= 1 - \pi_2$	
	$\tau_{11} + \tau_{12} + \tau_{21} + \tau_{22} = 1$			

Sample	$V = v$		$V \neq v$	
$U = u$	O_{11}	+	O_{12}	$= f_1$
$U \neq u$	O_{21}	+	O_{22}	$= N - f_1$
	$= f_2$		$= N - f_2$	
	$O_{11} + O_{12} + O_{21} + O_{22} = N$			

The simplest form of inference are direct ("point") estimates for the probability parameters, which are known as **maximum-likelihood estimates** (MLE). MLEs for π , π_1 , and π_2 are given by the *relative* joint and marginal frequencies p , p_1 , and p_2 as shown below.

$\pi = \tau_{11}$ $\pi_1 = \tau_{11} + \tau_{12}$ $\pi_2 = \tau_{11} + \tau_{21}$	$p = \frac{O_{11}}{N}$ $p_1 = \frac{R_1}{N} = \frac{O_{11} + O_{12}}{N}$ $p_2 = \frac{C_1}{N} = \frac{O_{11} + O_{21}}{N}$
true relative frequencies (population)	observed relative frequencies (sample)

Unfortunately, the difference between the estimate (sample) and the true value in the population can be very high (**sampling error**, especially when the observed frequencies are small) so this method is not applicable. We need to test whether the differences between the observed and the estimated values are due to the chance or if there is indeed some statistically significant relation. Assessing if we face the

chance event, the test of the null hypothesis is applied.

The null hypothesis of independence basically states that there is complete absence of association: statistical **independence**. When a pair type $(\mathbf{u}, \mathbf{v})$ has no association, the events $\{U = \mathbf{u}\}$ and $\{V = \mathbf{v}\}$ must be independent, which leads to the **null hypothesis of independence** H_0 below.

$$H_0: \pi = \pi_1 \cdot \pi_2 \approx p_1 \cdot p_2$$

The null hypothesis of independence stipulates a relation between the probability parameters (namely, $\pi = \pi_1 \pi_2$), but the parameter values are not completely fixed, and neither is the sampling distribution. The mathematical analysis is greatly simplified (and often made possible in the first place) when H_0 is reduced to a **point hypothesis** by inserting maximum-likelihood estimates for the probability parameters π_1 and π_2 . Most statistical independence tests for contingency tables are based on this point hypothesis.

The expectations $E_{ij} = E[X_{ij}]$ of the contingency table cells under the point null hypothesis of independence can easily be computed from the observed row and column totals. Values $E_{11}, \dots, E_{22}$ are referred to as **expected frequencies** and hypothesis tests that are based on the point null hypothesis can be understood as a comparison of the contingency tables of expected and observed frequencies, as schematised below.

	$V = v$	$V \neq v$	
$U = u$	$E_{11} = \frac{R_1 C_1}{N}$	$E_{12} = \frac{R_1 C_2}{N}$	O_{11} O_{12} $= R_1$
$U \neq u$	$E_{21} = \frac{R_2 C_1}{N}$	$E_{22} = \frac{R_2 C_2}{N}$	O_{21} O_{22} $= R_2$
			$= C_1$ $= C_2$ $= N$
expected frequencies			observed frequencies

When the observed data are 'unlikely' in a sense of being outside the range of values predicted by the probability distribution function built from the expected frequencies, the null hypothesis is rejected. This procedure is called a statistical **hypothesis test**. The same approach can also be used to obtain **confidence interval** estimates for the true values of population parameters.

IV.4.4.3 Association measures

An association measure is a formula that computes an association score from the frequency information in a pair type's contingency table. This score is intended as an indicator of how strong the association between the pair's components is, correcting for random effects. (Evert 2005, 75)

The cornerstone of the cooccurrence search is statistical analysis using association measures. In computational linguistics, various association measures were suggested and evaluated over the course of years (for an excellent overview of the past research, see (Evert 2005)). Two groups of association measures exist, those that measure:

- a) significance of association
- b) degree of association

The first group tests the existence of a link between the constituent elements. This test is conducted using statistical tests against a null hypothesis of independence and the final results show whether there is enough evidence for rejecting the null hypothesis of independence (no statistically significant relationship). This estimate can be expressed in the form of a p-value⁴⁰ and this value can be used to compare the results of the association measures inside the category.

The significance of association tells us whether the cooccurrences can be explained as random events, brought about by a pure chance, or whether there is certain, statistically significant link between the elements. The association measures exist in two variants: one-sided and two-sided depending on whether they distinguish between positive and negative association⁴¹. In general terms, the negative score means that cooccurrences are present *less* often than predicted by the null hypothesis, the positive score means the cooccurrences are present *more often* than if they were independent.

For one-sided association measures, high scores indicate strong positive association. Low scores (including negative scores) indicate that there is no evidence for a positive association (which could mean either that the components are independent or that there is negative association). For two-sided association measures, high scores indicate near-independence, regardless of their sign. A two-sided measure whose scores are always non-negative can easily be converted into a one-sided measure: for any pair type with negative association by multiplying the association score by -1. Thus, positive score indicate positive association and negative scores indicate negative association. The absolute value of the score depends on the association strength, with values close to 0 indicating near-independence. But as (Evert 2005, 76) found, for small absolute values, the distinction between positive and negative association is unreliable because of random effects. Such cooccurrences should be interpreted as “roughly independent”, with no clear evidence for either positive or negative association.

⁴⁰ Statistical hypothesis tests compute the total probability of all possible outcomes that are similar to or more “extreme” than the observed contingency table. This total probability is called a p-value. When it falls below a certain threshold, the sample is said to provide significant evidence against the null hypothesis, which is then rejected. Thus, the p-value provides a measure of the amount of evidence against H₀.

⁴¹ The terms one-sided and two-sided are parallel to the one-side and two-sided tests from the statistical theory.

The second group of measures is concerned more with the degree of association than with the amount of evidence supporting it. They are usually used for sorting the candidates into the list of the strongest pairs and selecting the first n-best cooccurrences.

Evert conducted a detailed review of the different association measures, taking as a base line Fisher's exact test. "After decades of controversy, most experts seem to agree now that Fisher's test produces the most meaningful p-values (cf. Yates 1984). We can thus take the Fisher association measure as a reference point for the significance of association group." (Evert 2005, 110)

$$Fisher = \sum_{k=O_{11}}^{\min\{R_1, C_1\}} \frac{\binom{C_1}{k} \cdot \binom{C_2}{R_1 - k}}{\binom{N}{R_1}}$$

The Fisher's exact test is however computationally expensive and it is advisable to use a different alternative. Mathematicians already computed how well various tests approximate the exact Fisher's test value, however Evert repeated their tests using the corpus that has the type of distribution found in natural language – ie. large sample size but highly skewed contingency tables where O_{11} is very small and O_{22} is extremely large.

From the comparison, the best results were obtained using the Poisson test:

$$Poisson = \sum_{k=O_{11}}^{\infty} e^{-E_{11}} \cdot \frac{\binom{E_{11}}{k!}}{k!}$$

Using this test also has several advantages, besides being computationally more efficient. It does not assume that the underlying data comes from the standard distribution. Poisson distribution is used for any elements (even rare) in which we predict a standard rate of data points appearance.

This test is therefore selected for SEMAN to serve as a filter that distinguishes statistically significant data from insignificant noise. But it is not yet enough, because in the corpus of documents we will find evidence of many relations. In order to focus the attention on the most important, we have to somehow sort and select only the best cooccurrences.

"With the wide range of association measures available, some guidance is needed for choosing an appropriate measure to be used in an application of cooccurrence data. While the theoretical discussion of Chapter 3 has helped to narrow down the number of options by grouping similar measures together, it cannot provide a definitive answer. The significance of association is a meaningful and well-defined concept, and Fisher's exact test is now widely accepted in mathematical statistics as the most appropriate quantitative measurement of this significance. The log-likelihood association measure gives an excellent approximation to the p-values of Fisher's test and has convenient mathematical and numerical properties. Consequently, it has recently become a de facto standard in the field of computational linguistics for the purpose of measuring the statistical association between words or similar entities." (Evert 2005, 137)

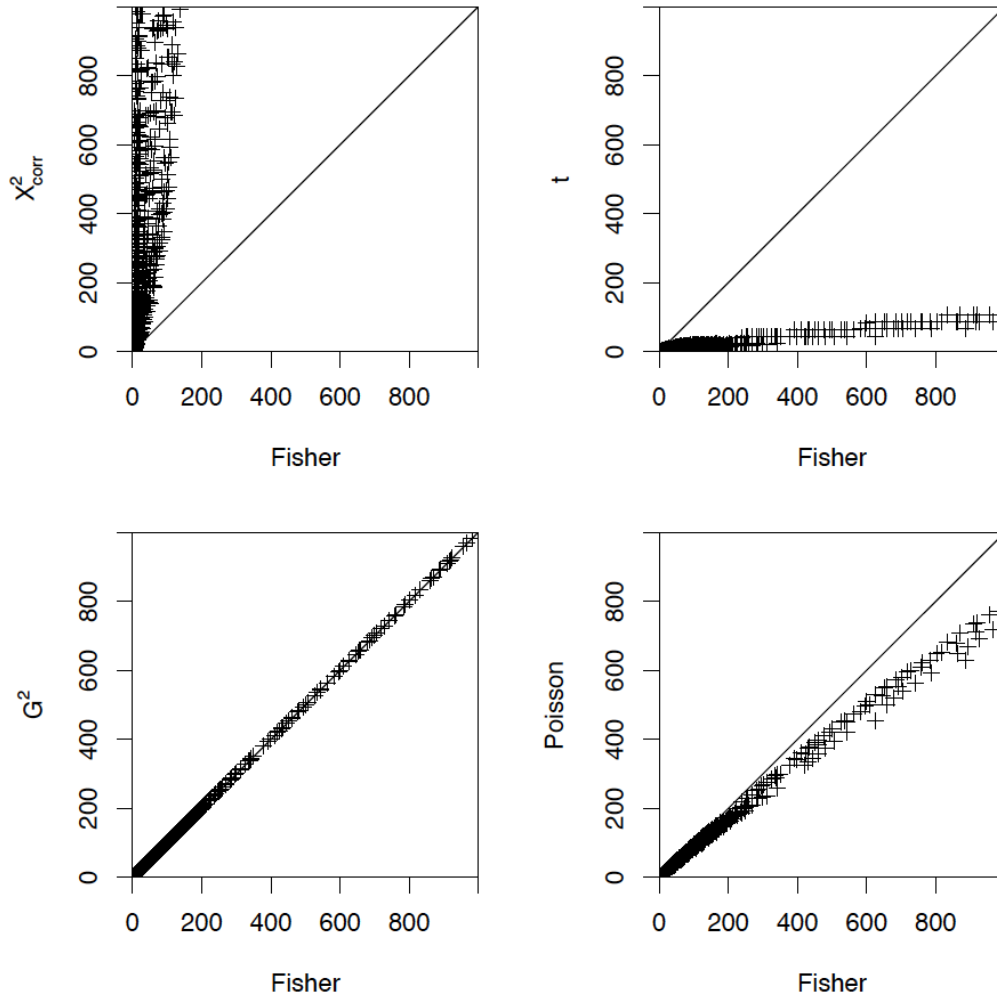


Illustration 19: Comparison of p-values for measures from the significance of association group, using Fisher as a reference point (labels on the axes refer to $-\log_{10} p_v$). Source: Evert, 2005, p. 111. G2 is the log-likelihood and “t” is the Students t-test. These graphs show that X2 statistic favored rare events, while the t-test gave undesired advantage to frequent events. The other two tests results were much closer to the Fisher's p-value.

Following this advice, the log-likelihood association measure was selected as the default measure for ordering cooccurrence pairs. It has the form:

$$\log - \text{likelihood} = 2 \sum_{ij} O_{ij} \log \frac{O_{ij}}{E_{ij}}$$

In the Everts formula, the logarithm is undefined when there are empty cells (with $O_{ij}=0$). For such cells, the entire term evaluates to zero (because $0 \cdot \log(0)=0$ by continuous extension) and can simply be omitted from the summation.

The Evert's version is in effect identical to the log-likelihood as invented by the original author Dunning:

$$\log - \text{likelihood}_{\text{Dunning}} = -2 \log \frac{L(O_{11}, C_1, r) \cdot L(O_{12}, C_2, r)}{L(O_{11}, c_1, r_1) \cdot L(O_{12}, C_2, r_2)}$$

$$L(k, n, r) = r^k (1-r)^{n-k}$$

$$r = \frac{R_1}{N}, r_1 = \frac{O_{11}}{C_1}, r_2 = \frac{O_{12}}{C_2}$$

The effectiveness of the log-likelihood association measure stems from the attention it gives to all the elements of the contingency table. The ranking is derived from the sampling distribution, the smaller the probability of the a sample outcome under the null hypothesis, the bigger is the “surprise” and therefore the probability that the difference is not due to a chance.

And because the log-likelihood statistic has an asymptotic χ^2 (chi-squared) distribution, we can directly compare its results with the χ^2 test statistics with one degree of freedom (because the 2x2 contingency table and one free parameter, O_{ij}). The log-likelihood measure is two-sided but it can also be converted into a one-sided measure by changing the sign of the test statistic when $O_{11} < E_{11}$, indicating negative association. The test statistics formula is:

$$\log - \text{likelihood}_{\text{ratio}} = -2 \log \frac{\max P(\vec{X} = \vec{O} | N \wedge \pi = \pi_1 \cdot \pi_2)}{\max P(\vec{X} = \vec{O} | N)}$$

When we search for the significant changes, we compare data coming from two collections of documents – we call them corpus 1, and corpus 2 C_1, C_2 using the following algorithm:

```
# retrieve set of used semantic codes
sem_components = get_sem_components(opened_index)

# find all possible collocations of terms, considering their index position
existing_collocations = get_collocations(field, distance_w)

# test statistical significance of collocation occurring/change/missing
for collocation in existing_collocations:
    token_a = collocation.get_term()
    token_b = collocation.get_incidental_term()
    signif_collocation = set()
    if token_a in sem_components and token_b in sem_components:
        for part_a in token_a:
            tf_a = get_term_freq(part_a)
            for part_b in token_b:
                tf_b = get_term_freq(part_b)
                signif_score = test_significance(part_a,
                                                part_b,
                                                tf_a, tf_b,
                                                tf_ab,
                                                alpha_level)

                if signif_score:
                    signif_collocation.add((comp_a, comp_b, signif_score))

if signif_collocation > 0:
    store.append(signif_collocation)
```

Following the procedure describe above, we can search for both relational and positional pairs of cooccurrences. The evaluation of this procedure will be given in the chapter V.2.3 Search for significant combinations, p. 142

IV.4.5 Dictionary creation and maintenance

One of the most difficult and time consuming operations for any content analysis research is the creation, and maintenance of the coding dictionary. This is probably the most expensive operation and the existence or absence of a 'good' coding scheme bears a lot of influence. Because the dictionary is so important, even the best algorithm or methodology cannot relinquish mistakes and ambiguities that were introduced in the representation of the knowledge. Systems that rely on the human created/supplied knowledge representation thus face the real problem of knowledge acquisition – the knowledge bottleneck. (Turmo, Ageno, and Català 2006)

When reviewing the existing approaches to knowledge representation tools, I have found many systems built specifically for the ontology maintenance.⁴² But most of them work with a different conceptualization of the dictionary and are not appealing for the nature of the USL.⁴³ Especially because there are many links inside the semantic network, and many of the tools are either too complex or built for the maintenance of the hierarchical systems where the graphs are acyclic. But an entry in the USL may have many parents and exhibit a sort of cyclic relationships – either because of the mistakes, or simply by the complex nature of the world of knowledge representation.⁴⁴

Also the two content analysis tools presented provide some help with the dictionary build. In many other cases, we can find a sophisticated graphical user interfaces with greater control and much advanced functions. However, the experience of V. Smetacek spoke in favour of keeping it simple and lean. That advanced GUI tools often stand in the way of effective and fast dictionary administration. After all, the TABARI system itself is a good example of the same approach. The dictionary in TABARI is edited via set of commands, very quickly and efficiently once analysts are familiarized with the keyboard shortcuts.

Undeniably, the command line tools may be much faster, but sadly, they can be rather unfriendly and confusing when they do not provide enough context. In the case of TABARI, the structure of the dictionary does not exhibit such a multi-relational nature as

42 For a list of some, see: http://semanticweb.org/wiki/Category:Ontology_engineering_tool [Accessed: 24-7-2011] which lists SMW+ (<http://semanticweb.org/wiki/SMW%2B>), Semantic Turkey (<http://semanticturkey.uniroma2.it/>), RDF2GO (<http://rdf2go.semweb4j.org/>), NeOn Toolkit (http://neon-toolkit.org/wiki/Main_Page), PoolParty (<http://poolparty.punkt.at/>). But we should also not forget Protégé (<http://protege.stanford.edu/>), which was needless to say, the only tool widely spread at the time when the work on SEMAN started.

43 And in the words of the supervisor of this work: the more and fancy approaches were tried before, but in the end, the plain old representation of the dictionary is the best. So it was kind of a requirement to keep the form of the USL untouched.

44 Another requirement, with which I cannot disagree, is to build the dictionary in as simple ways as possible. But not simplistic.

for USL, and it would be very hard to present it in the console environment. The graphical interface can provide much more information visually, and provides certain views more easily.

Analysts need a way to view the whole semantic network, edit not only single records, but also whole parts of the dictionary or even everything. Change and automatically propagate changes to the whole network. For those reasons, it was decided to implement a hybrid approach between a very quick and familiar mode of the text editing software, together with the advanced characteristics of widget based GUIs. In a sense, it is the mix of the proven interface of the TABARI system, where analysts control all the operation via command line shortcuts, and the fully functional graphical interfaces of word processing editors on the other hand. Because we do not want to miss the advanced features and services of the graphical interface. Therefore, in the domain of the dictionary, we have opted for an advanced editor interface rather than advanced graphical interface to dictionary.

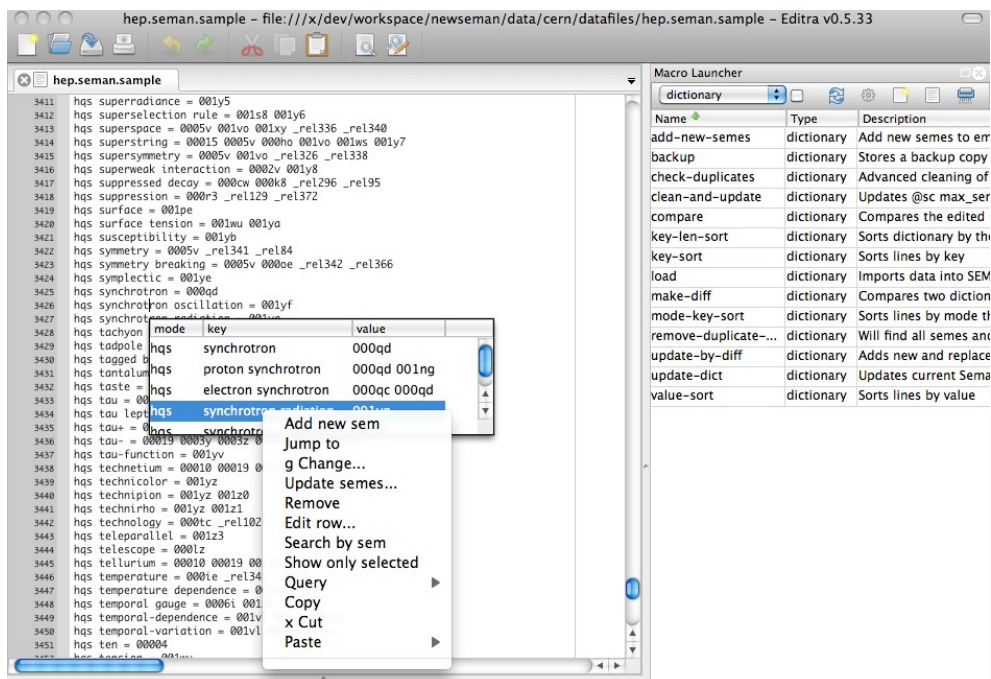


Illustration 20: The dictionary editing session, with some special functions activated (autocompletion of semantic codes)

A special editing mode is built inside the programmers editor called Editra. This component is an open-source text editor, with programmable features, and written in Python. It gives users control of the dictionary maintenance but also provides complete control over SEMAN, thus it naturally becomes a working environment not only for the dictionary maintenance, but also for the analysis and the control of the coding operations, as will be show later.

As can be seen from the screenshots, the dictionary can be edited as a normal text file. Each record is listed on a separate line and there is a simple format of:

<code> <pattern> = <categories>

Almost every operation can be executed using only keyboard shortcuts, which makes it analogous to the TABARI fast editing mode. Users do not need to learn commands and in most cases they will be already familiar with the concept of a text editors. The editor hides the complexity of the operations behind the scenes – it can check for duplicates, validate the entry form, help users to search for codes already used, link to them, autocomplete the lexical entries, search for other similar entries or provide statistics on the usage. These operations are in many cases context dependent, and we will not list them here in their completeness.⁴⁵

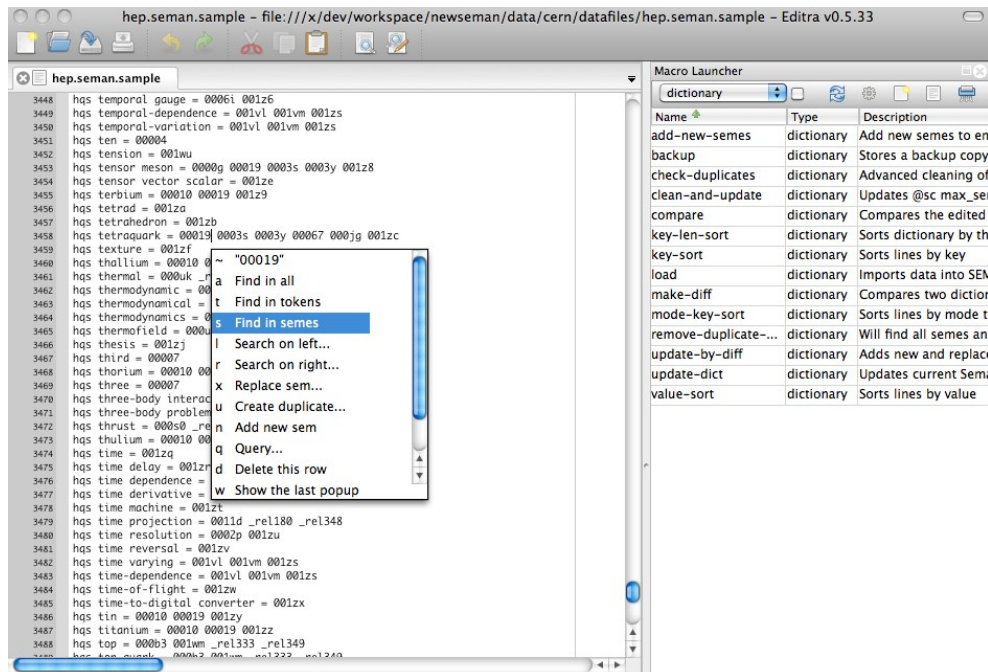


Illustration 21: Another set of commands of the editor; on the right side are visible the maintenance routines for more complex operations

With the additional widgets, it is possible to edit and maintain the dictionary in an effective manner. Even big dictionaries, consisting of hundreds of thousands of records – as was the case of the converted version of Wordnet, that was transformed into the form of USL. This dictionary had more than 60 thousands entries, not strictly hierarchical. If we were to use the simple 'search and replace' functionality of normal text editors, it would be cumbersome, for example, to find all items that share a few semantic codes, choose from them a subset of entries, and replace this subset with a newly defined categories. With SEMAN, these operations are simple.

Besides the special functions of the editor available in the text editing area, there are also some special routines for dictionary maintenance, consistency checking, sorting, comparison with other (older) versions of the dictionary and similar. For example to edit the dictionary, view only the changed parts and import only the differences between the versions – or directly apply the new version of the dictionary on the texts.

This is perhaps not important when we need to edit a few categories, such as in the case of Yoshikoder, but it is very much needed when we do global, bulk updates to the whole parts of the dictionary, changing hundreds and thousands of records. The experience shows that the editor must be very flexible and at the same time simple. Because of these reasons, the

⁴⁵ They can be found in the appendix of (Schrodt 2009)

editing mode was tailored to the concept of the programmable editor. And this makes it possible to write and execute specific scripting routines, perhaps only on part of the dictionary. A number of the functions were implemented, but the powerful combination of Python and the scripting components gives the user the option to write her own processing routines. The Python language is ideal for such operational mode.

IV.4.6 SEMAN GUI and scripting control

The same interface which is used for dictionary maintenance is used also for controlling the execution of analysis in the interactive mode. In the series of screenshots below, the panel on the right side contains various processing steps that the user can apply on the

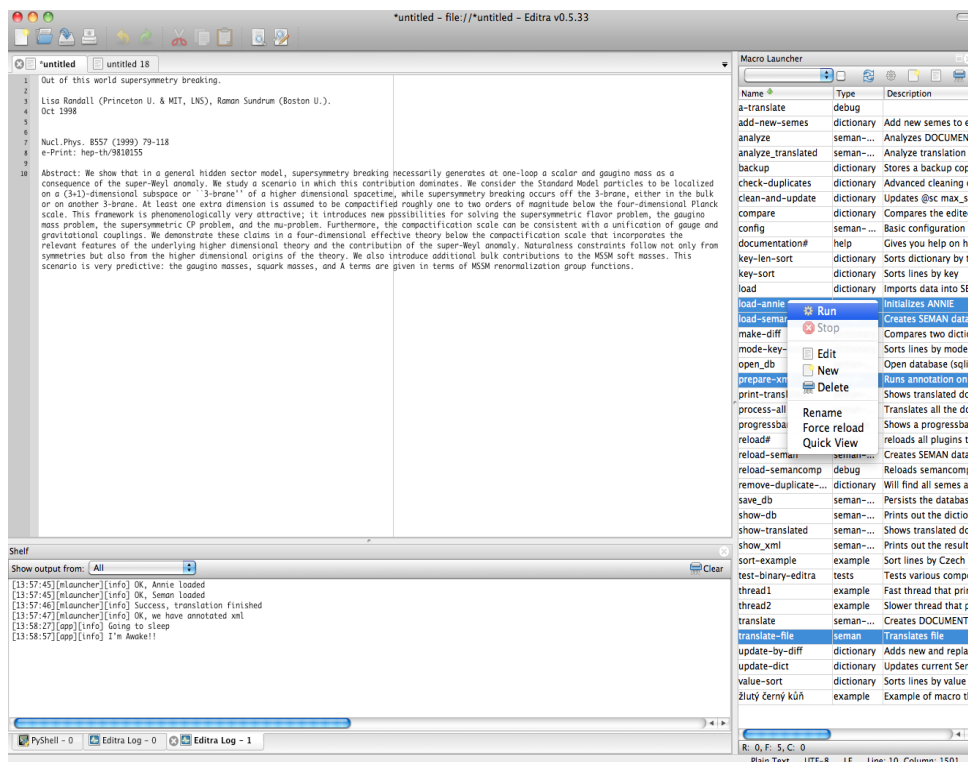


Illustration 22: Example debugging session of the document analysis - on the right side a few processing steps are selected and executed as one. The window at the bottom shows log messages as the analysis proceeds.

individual document or the collection of documents. These steps were described in the previous chapters.

Another feature that distinguishes SEMAN from most of content analysis software is the programmatic interface. While all of content analysis packages contain prepared methods of processing (which SEMAN also has), none of them give researchers tools to write ad-hoc analytical routines, just to explore the data. But SEMAN provides such functionality. For example, if the user wants to tests the null hypothesis that there exists no dependence between word pairs, but instead of the standard log-likelihood or a pearson test, she would like to apply the risk ratio or information gain test, all that is needed is to write one-purpose macro without any need to compile or rebuild individual components.

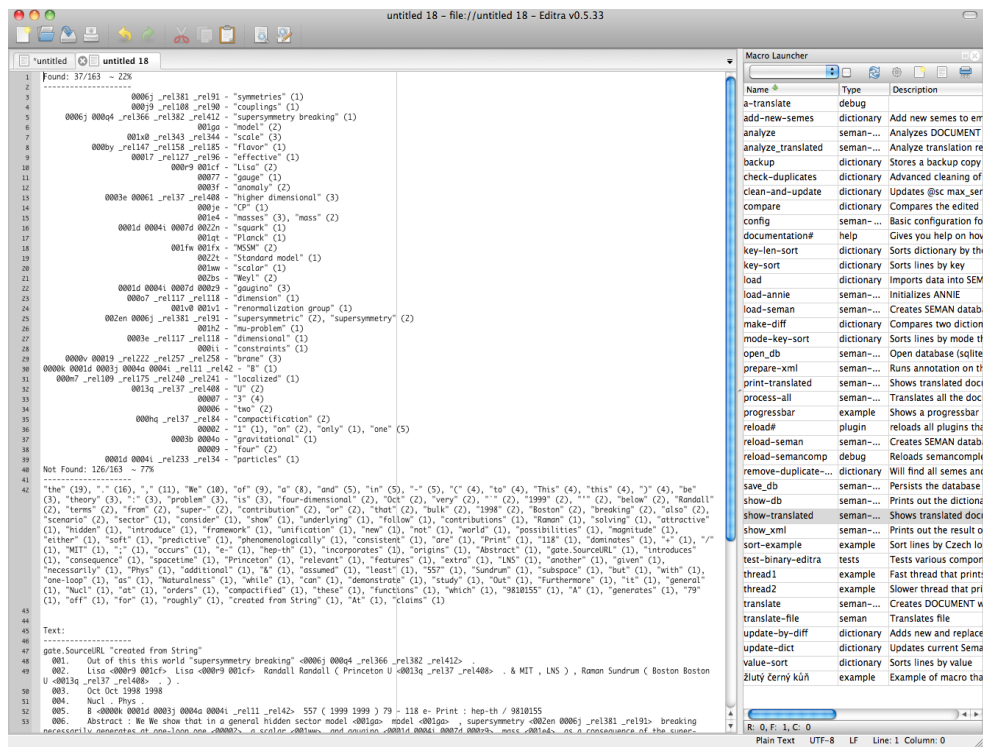


Illustration 23: Example document analysis, showing which tokens were found - with their absolute frequencies.

In fact, more than a one-purpose tool for content analysis, SEMAN is also a content analysis environment. Power users, who are comfortable with scripting, do not need to leave the Editra interface. They can open the interactive console and test their own methods directly from inside the editor. All the methods and packages of SEMAN are available. Windows and Macintosh users may carry such environment even on the portable storage media, having the full functionality including the NLP components.

The screenshot shows the Editra IDE interface. The main editor displays a Python script named `macro-9.py` which implements a Risk Ratio test. The script includes imports for `find_composites`, `config`, `log`, `sqrt`, and `exp`. It defines a `run` function that takes a text corpus and parameters, and a `rr_test` function that calculates the Risk Ratio and its confidence interval. The script also includes a call to `find_composites.run` with various configuration options.

On the right side, the Macro Launcher panel is visible, showing a list of macros with their names, types, and descriptions. The macros include `key-len-sort`, `find-collocations`, `a-translate`, `compare`, `documentation#`, `progressbar`, `reload#`, `sort-example`, `thread1`, `thread2`, `žlutý černý kůň`, `analyze-translated`, `open_db`, `reload-semancomp`, `show-db`, `show-translated`, `show_xml`, `reload-seman`, `add-new-semes`, `check-duplicates`, `clean-and-update`, `load`, `remove-duplicate-...`, `save_db`, `backup`, `make-diff`, `key-sort`, `value-sort`, `mode-key-sort`, `update-by-diff`, `update-dict`, `print-translated`, `process-all`, `config`, `analyze`, `load-annie`, `load-seman`, `prepare-xml`, `translate`, `translate-file`, and `test-binary-editra`.

At the bottom, the PyShell window shows the execution of the script, displaying the calculated Risk Ratio and its confidence interval.

Illustration 24: Risk Ratio is the statistical method implemented in Yoshikoder. It tests whether the difference between two documents are significant.

$$CI = \log(RR) \pm SE * z_{\alpha/2}$$

This illustration shows how easy it is to add the same functionality to SEMAN. We can test different test functions, the shell provides a full environment and is interactive.

Yet it would be mistake to conclude that the portable nature is important only in the context of individual users. Text processing of a large corpus is often executed on the grid or in the cloud, in heterogenous computer architectures, with different Python versions, or operating systems, so it is vital that content analysis tool can be used in such scenarios.⁴⁶

⁴⁶ At the time of writing, the CERN Lxplus network is a cluster of 30 thousand computer nodes made available (on demand) to the HEP community. Machines are running either Scientific Linux version 4 or version 5 operations systems. This distribution is based on the Fedora operating system. SEMAN routinely runs in this environment as well as on Windows NT, Linux Ubuntu and Macintosh OS X machines.

IV.5 COMPARISON

It might be interesting to look more than one decade back at a review of artificial intelligence methods and their use in content analysis studies. It was found in 1998 that the CA tools were well suited for instrumentation type studies – i.e. those in which we are not interested in obtaining the meaning, where the 'intended' meaning is not a component of the message that should be extracted. The review identified 4 main problem areas where content analysis approaches failed: (1) parsing – discovering the syntactic structure of the sentences; (2) context – deciding the meaning of sentences when this depends upon other, surrounding text; (3) prior knowledge – assumptions about the world outside the of a discourse that are essential to its meaning; (4) semantic variability – differences in the meanings of words and sentences from one speaker to another. (Cuilenberg, Kleinnijenhuis, and de Ridder 1998) The authors reviewed the chosen approaches, but after a lengthy discussions, concluded that the analysis at the level of interpretation of complex texts is still a long way away.

The report was issued in 1998 and in the meantime at least part of the landscape changed. The most striking example is the development of computer software for application of the Gottschalk-Bechtel method. This method measures psychological traits and is used in clinical studies to assert different levels of anxiety, hostility, hope and 9 other psychological states. The software used for content analysis of the transcripts has to deal with very complex and incomplete sentences, but it was shown that the reliability is comparable to human coders and the software is even able to achieve higher levels than 0.8 accuracy. (Gottschalk and Bechtel 1995)

However, this is still an isolated problem on which a team of researchers worked since 1982 and there exists an extensive theory behind the scale. For other areas of ambiguous general-domain content analysis, we are still not in the phase where content analysis software could parse and interpret complex syntactical and semantical structures.

Nevertheless, considerable development has taken place, at least for the problem (1), the syntactic structure and parsing of sentences we can report that current techniques are able to achieve close to 90% accuracy (Cer, Marneffe, Jurafsky, and Christopher D. Manning 2010). As for the disambiguation of words (3) the present content analysis tools still rely more on knowledge representation maps (the dictionaries) than on the methods of automatic word sense disambiguation⁴⁷ And the other two remaining areas are mostly untouched. Interpretation of sentence meaning is a very hard problem (as complex as word sense disambiguation, AI complete) but its solution is not needed for tasks of the classical content analysis. What would be more interesting are the developments in the field of automatic knowledge acquisition, creation of knowledge representations. Yet most tools, including SEMAN, still rely heavily on manually acquired and curated dictionaries which is perhaps their biggest weakness. It is fair to say that all these tools belong to the same paradigm as the knowledge representation systems that were developed until late 1990's before the statistical learning and automatic knowledge extraction systems took over (Turmo, Ageno, and Català 2006).

⁴⁷ The following chapter will contain information about an experiment where we compared results of classification using automatically disambiguated word senses and heuristic rules. Surprisingly, the heuristic rules of word sense disambiguation may be enough for many content analysis studies.

But if we stay in the domain of classical content analysis, we can compare the systems – or better, we can compare the different approaches that the four systems represent. In the nature of processing, they are all very similar. There is a dictionary of patterns in the core of the system – either constructed manually, or derived from the other existing dictionaries and knowledge organisation databanks. Texts are parsed, coded and recognized entities analysed or exported for further analysis in external programs.

The main difference of SEMAN from the majority of content analysis tools is the application of USL – while the other content analysis programs are designed to find and assign indicators into (single or multiple) categories, in the case of SEMAN, the coding scheme has the nature of a combinatorial classification system. Not exactly hierarchical, but similar in functionality to the hierarchical classification systems such as DDC or UDC.

Albeit codes can be used as unique identifiers of a category exactly in the same way as classical content analysis tools, they can also be combined with the aim of finding relations. For example, the classical tools allow analysis of the distribution of signals across the categories, SEMAN also answers questions like: What is the proportion of analysed documents, that belong to the category with features X and Y and Z? Are there some significant connections between a feature X and Y?

In order to answer these questions, SEMAN is taking advantage of the network of relations that is encoded inside the dictionary. Operational mode is similar to other tools. First a conceptualization of the research questions must be encoded in the dictionary of categories. For that purpose, SEMAN provides its own graphical user interface. Directly as the user edits the dictionary, SEMAN can analyse texts using the new version of the coding scheme and provide feedback. The user controls content analysis operations directly from the same interface which is used for the dictionary maintenance and debugging. Coding operations can be executed for individual files or whole collections of files. Results are then stored in the index or database. Or exported in various formats for further analysis.

If we put aside the nature of the USL, the biggest difference is that SEMAN can handle very complex workflows and can employ a wide range of NLP procedures. It is not simply a find-and-count tool. In the default configuration SEMAN disambiguates meaning using POS of the extracted tokens, but it can also work similarly to TABARI – by using a full sentence parsing with much more powerful options. To that effect, internally SEMAN does not work with text, but with acyclic graph of tokens. However in the present state, the TABARI with its pattern matching mechanism is easier to use and allows for more granular and semantically correct matching. For SEMAN, other possibilities are open but they require programming skills and are not part of the dictionary definitions.

The third difference comes with the open nature of the application. It contains a standard set of tools of content analysis, but technically savvy users can modify them and obtain a very specific results with small changes. SEMAN works as a content analysis tool, but it is open and offers the previously mentioned workflow engine and many text-analysis components. Optional modules may be plugged into the workflow(s), some of which were described - such as word sense disambiguation, format conversions, keyword extraction. With this apparatus it is possible to prepare specific processing tailored to the nature of texts, as will be illustrated in the third section of the dissertation.

Finally, we should mention also the programming language SEMAN is written in. At the beginning of the project it was not clear what programming language to choose, but with time it became clearer that Python was a very good choice. There are many strong arguments for it, mainly its ease of use, rapid prototyping, nature of the scripting language, weak typing and very clear syntax and readability of the code. The main parts of the system are thus written in Python, but this does not limit SEMAN only to Python. The language serves very well as a 'glue' between other components, and where the speed or extendability is an issue, we can choose to use compiled languages.⁴⁸ But the Pythonic nature may have influenced the final form of the system. Now, it can be very useful for power users to whom it exposes advanced functionality easily. But it is not as usable and straightforward as Yoshikoder, even if the GUI does make SEMAN more accessible.

⁴⁸ So SEMAN now incorporates components written in C++ and Java, but from the viewpoint of the user they are invisible.

V. EVALUATION OF SEMAN

In this section we shall focus on SEMAN; a system that was designed to work with USL. A few experiments were designed to show capabilities of the application. First part of the evaluation is concerned with a pattern matching mechanism, then we examine results of translation in the case of document classification and finally we look at the search for concept cooccurrences. The methodology and corpora will be described in respective chapters.

V.1 THE PATTERN MATCHING MECHANISM

The pattern matching mechanism present in SEMAN is different from the majority of content analysis tools. It does not work with regular expression or wildcard expressions. Instead the search uses sets of possible combinations of prefixes, suffixes and radices, the search is deterministic, based on the entries present in the dictionary. However, the nature of the combinatory principle poses certain challenges (as well as opportunities) and the mechanism must work well should the system produce useful data. The extraction is therefore an important stage and the matching mechanism is a crucial component. We will look at its speed and accuracy in comparison with another tool that does a similar job but works differently.

V.1.1 BibClassify

BibClassify is an automated keywords extractor developed by CERN. It performs an extraction of keywords based on the recurrence of specific patterns that are listed in a controlled vocabulary. The controlled vocabulary is a thesaurus of all the terms that are relevant in the field of High Energy Physics.

BibClassify accepts thesauri in two formats, either as a simple text of allowed keywords with one keyword per line, or in a XML format of RDF SKOS. Out of the two, only the second form is suitable for recording the taxonomic relationships between concepts. This is

a richer and more complex structure to describe concepts.¹ In RDF SKOS, every keyword is marked with a tag `concept` which encapsulates the full semantics and hierarchical status of a term - including synonyms, alternative forms, broader concepts, notes and annotations - rather than just a plain keyword. For example:

```
<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#asymmetry">
  <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.yieldasymmetry"/>
  <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.timeasymmetry"/>
  <composite
rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.timereversalasymmetry"/>
  <composite
rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.supernovaasymmetry"/>
  <prefLabel xml:lang="en">asymmetry</prefLabel>
  <hiddenLabel xml:lang="en">/asymmetr\w*/</hiddenLabel>
  <hiddenLabel xml:lang="en">/nonsymmetric\w*/</hiddenLabel>
</Concept>

<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#asymptoticbehavior">
  <composite
rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.transformationasymptoticbehavior"/>
  <composite
rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.totalcrosssectionasymptoticbehavior"/>
  <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.space-
timeasymptoticbehavior"/>
  <prefLabel xml:lang="en">asymptotic behavior</prefLabel>
  <altLabel xml:lang="en">asymptotic behaviour</altLabel>
</Concept>

<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#ATLAS">
  <prefLabel xml:lang="en">ATLAS</prefLabel>
</Concept>

<Concept rdf:about="http://cern.ch/thesauri/HEP.rdf#atmosphere">
  <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.nucleusatmosphere"/>
  <composite
rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.neutrinoatmosphere"/>
  <composite rdf:resource="http://cern.ch/thesauri/HEP.rdf#Composite.muonatmosphere"/>
  <prefLabel xml:lang="en">atmosphere</prefLabel>
  <hiddenLabel xml:lang="en">/atmospher\w*/</hiddenLabel>
</Concept>
```

The thesaurus is actively maintained and enriched by subject specialists and BibClassify exploits the richness of the thesaurus to produce accurate results.

In its basic form, BibClassify selects keywords from a fulltext document based on the frequency of thesaurus terms in it; by calculating how many times a keyword from the thesaurus (and its alternative and hidden labels) appeared in the text. Results are ranked accordingly. This simple term/document frequency ranking makes it easier for us to compare performance of the pattern matching. However, the systems differ considerably and BibClassify is not a simplistic token matcher. For example when selecting keywords, it gives preference to identified groups of tokens and filters out concepts which are not

¹ The specification of the SKOS language and various manuals that aid the building of a semantic thesaurus can be found at the SKOS W3C website. Furthermore, BibClassify can function on top of an extended version of SKOS, which includes special elements such as keychains, composite keywords and special annotations.

marked as “core concepts” or which are marked as “no standalone”². The thesaurus thus effectively contains simple disambiguating routines that work well for the specific subject domain.

It differentiates between single and composite concepts. In the majority of cases single concepts are made of single keywords. Nevertheless the difference is semantic. Humans tagged certain words or groups of words as single concepts, and these can be present in combinations with others. For simple concepts, one or more regular expression patterns are compiled and when compiling the regular expressions around the candidate terms, the basic rule is:

`(?:[AA-Za-z0-9\+-]) ( + candidate term + ) (?:[AA-Za-z0-9\+-])`

The word separator (in bold) differs from the standard regular expression for non-whitespace character (`\s`) as it includes plus and minus signs (that in the case of HEP thesaurus terms cannot be regarded as whitespace). When compiling the regular expressions, BibClassify also performs a number of transformations. For instance it creates multiple patterns that match form variants, for illustration, if a word ends with following patterns:³

```
"[^e]ed$"
"ics?$"
"[io]s$"
"ium$"
"less$"
"ous$"
```

It will not be searched in the case insensitive manner, but only first letter uppercase or whole expression lowercase will be allowed. So the candidate term `electronics` is converted into `[Ee]lectronics?$`. In the case of other patterns, the search will be case insensitive. There are also a number of patterns that deal with different word spellings, for example:

term	becomes:
"color"	<code>r"colou?rs?"</code>
"colour"	<code>r"colou?rs?"</code>
"deflexion"	<code>r"defle(x ct)ions?"</code>
"flavor"	<code>r"flavou?rs?"</code>
"flavour"	<code>r"flavou?rs?"</code>
"gas"	<code>r"gas(s?es)?"</code>
"lens"	<code>r"lens(es)?"</code>
"matrix"	<code>r"matri(x(es)? ces) "</code>

Certain invariable forms will be left unchanged if they contain patterns such as: "any", "big", "chi", "der", "eta", "few", "low", "new", "non", "off", "one", "out", "phi", "psi", "rho", "tau", "two", "van", "von", "hard", "weak", "four", "anti", "zero", "sinh" and many others.

² I.e. those that must be accompanied by another concept

³ We do not list all the BibClassify patterns here.

Also during the compilation of patterns, BibClassify will check for UPPERCASE candidate terms, they will be considered acronyms and patterns are always case sensitive. Similarly if the candidate term contains a digit, the pattern is unchanged – so `rho(1980)` will be searched as case insensitive `rho(1980)` but if there was no digit, the original pattern would have been changed in number of ways. This exemplifies the complex nature of the regular expressions automatically derived from the taxonomy.

We should also not forget that the original candidate words may be changed to mimic stemming, so for example if the word ends with the following patterns, the final pattern will be changed to contain the expression from the second column:

"ional"	"ional(ly)?"
"([ae])n(ce t)\$"	"\ln(t ces)?"
"og(ue)?\$"	"og(ue)?s?"
"([^aeiouyc])(re er)\$"	"\l(er re)s?"
"([aeiouy])[sz]ation\$"	"\l[zs]ations?"
"([aeiouy])[sz]ation\$"	"\l[zs]ations?"
"([^aeiou])(y ies)\$"	"\l(y ies)?"
"o\$"	"o(e?s)?"
"(x sh ch ss)\$"	"\l(es)?"
"f\$"	"(f ves)?"
"ung\$"	"ung(en)?"
"([^aiouy])s\$"	"\ls?"
"([^o])us\$"	"\l(i us(es)?)?"
"um\$"	"(a ums)?"

All the various adjustments are done to patterns of the single keyword entries but because also their combinations are matched, BibClassify will sometimes use special regular patterns or it will explore the immediate context and search for combinations of the components up to a certain distance around single matches. And only certain patterns are considered as valid keyword separators, such as: "of", "of a", "of an", "of the", "of this", "of one", "of two", "of three", "of new", "of other", "of many", "of both", "of these", "of each", "is". If different tokens separate keywords, the matches will be rejected.

And finally we should mention that BibClassify will normalize the fulltext before processing – for example Greek alphabet characters that are used for mathematical expressions are replaced with their ascii names such as Sigma, or Lambda and dashes in the end of lines indicating word split will be merged. Citations and references to other papers will be removed from extraction. Empty spaces normalized and so on. As we can see, BibClassify contains a number of rules that were discovered with time and that aim at eliminating false positives and improve precision. It is a careful matching mechanism represented by the regular expressions automata and against this system the performance of SEMAN will be compared.

V.1.2 Comparison

The corpora used in this task is made of 2084 fulltext documents randomly selected from the six physics related fields of the arXiv.org digital library.⁴

The six groups are:

ID	arXiv codename	Name
0	astro-ph	Astrophysics
1	hep-ex	High Energy Physics - experimentation
2	hep-lat	High Energy Physics - lattices
3	hep-ph	High Energy Physics - phenomenology
4	hep-th	High Energy Physics - theory
5	math-ph	Mathematics - phenomenology

All the papers were drawn from a pool of documents classified by the arXiv section manager. They were given one main subject and possibly several additional categories. Such a multi-class corpus better represents the distribution of documents in a real life situation.⁵

HEP corpus distribution						
.	0	1	2	3	4	5
0	378	0	0	3	3	0
1	0	365	2	6	1	0
2	0	2	351	13	10	1
3	3	6	13	351	3	0
4	3	1	10	3	334	8
5	0	0	1	0	8	305

Table 2: Distribution of files in the HEP corpus categories, the count in the diagonal shows the total number of documents in the category, the number outside the diagonal shows number of multi-class documents. For example there are 378 items in total for Astrophysics, 3 of them are also HEP phenomenology and 3 are HEP theory papers

For purposes of comparison, texts were normalized using routines present in BibClassify. Both BibClassify and SEMAN will use the same taxonomy with 3244 single and 56161 composite concepts. By composite, we mean a concept that is made of a combination of simpler keywords. For the 3244 single keywords BibClassify creates 4167 regular

⁴ The corpus files as well as the code are distributed together with SEMAN so that the measurements can be repeated.

⁵ The classification experiments with a corpus of single-category documents showed very high precision. I.e. the job of classification was too easy. As such a situation is highly improbable in normal circumstances the corpus was drawn from multiple-class documents.

expressions. The number is higher because many concepts contain additional regular expressions. For the other composite keywords, there are 339 additional regular expression patterns – in total, BibClassify is working with 4506 compiled regular expressions which make for the rest of the 59405 entries.

While BibClassify reads the taxonomy from the RDF file, SEMAN generated it from the source files.⁶ The dictionary is generated automatically, without any human intervention. The entries are stemmed using the Porter stemmer, the original full form, and the stemmed and lowercased forms are both saved for further reference. The dictionary also contains all suffixes that were removed during stemming. The total number of entries for each category is shown below:

<i>Mode</i>	<i>Number of values</i>	<i>Explanation</i>
hsI	59405	Main keys
hsi	1085	Non-stemmed versions of synonyms
hsq	53082	Stemmed versions of the above
hss	165	Suffixes
total	113737	

This shows, that the number of entries in the dictionary of SEMAN more than doubled. If we look at the number of patterns by their composite type, we will see the following:

single keywords	7699
keywords with 2 components	93584
keywords with 3 components	12438
keywords with 4 components	16

Due to the conversion problems for some documents and the way in which documents were selected for the run (with the same number of documents in each category for both BibClassify and SEMAN; we also use only the training part of the corpus, i.e. 80%), the run included 1682 documents. On the machine with the following specification:

```
Model Name: MacBook Pro
Memory: 4 GB
Processor Name: Intel Core i7
Processor Speed: 2.66 GHz
Number Of Processors: 1
Total Number Of Cores: 2
L2 Cache (per core): 256 KB
```

⁶ In principle, there is no difference in the contents, only that some fields are not exported into the RDF form. However, these fields are not used by any of the tool, therefore can be ignored. For purposes of our comparison the dictionary is identical.

L3 Cache: 4 MB
 Python: 2.6, 32bit
 OS: Mac OS X 10.6

The two programs finished in the following time:

	BibClassify	SEMAN
Total time	9368.74s	1505.39s
Average time per document	5.57s	0.895s

The matcher such as BibClassify, is using regular expression patterns while searching for entries in the whole text. It must try all the expressions available. This explains the difference in speed of the two systems. If we were to increase the size of the dictionary the time of matching for BibClassify would increase linearly to the number of expressions. While for SEMAN the increase is sub-linear.

In the case of SEMAN most time is spent in matching the single keywords (which are groups of tokens of any length). As in this run we were not using any prefixes, the worst case would be *basic words x suffixes combinations*, i.e. $165 \times 7534 = 1.270.335$ combinations. If there were 10 prefixes, this number would increase 10 times, but it is not uncommon to work with several hundred prefixes. In such cases the number of combinations is getting close to one billion. So how can we explain the fact that SEMAN is still more than 6 times faster than the regular expression matching?

The answer lays partly in the algorithm and partly in the regularity of the natural language. The theoretical number of combinations wildly surpasses any probable number of combinations of prefix-radix-suffix combinations present in the real language. Because the pattern matching inside SEMAN follows local constraints, the number of examined combinations is a fraction of the theoretical combinations.

However we have to see whether the two systems are comparable in terms of precision before we can conclude anything. Maybe SEMAN matches only a few patterns and thus spends less time searching and can finish faster. Or the results are not useful. To verify it, we will see the comparison of the keywords extracted from the corpus by BibClassify and by SEMAN with three different configurations. BibClassify represents the baseline and only keywords that were matched by one or the other extractor are counted in the comparison (to illustrate, the total number of patterns is 59405, but slightly less than 1/5 were present in the testing corpus of 1682 documents).

In the case of SEMAN all tests were run on the corpus that was automatically POS tagged with ANNIE. R1 represents results extracted by SEMAN using the default automatically generated and stemmed dictionary. When there was an ambiguous pattern (i.e. token that several definitions) the first meaning was automatically selected. R2 improves on the selection of the ambiguous matches using the POS tags, i.e. matches on verbs are discarded, adjectives are considered in combination with nouns, definitions of verbs are not used for translation of homograph nouns etc. The R3 set contains results using the same POS corpus after the SEMAN algorithm was corrected to correspond more closely to the way BibClassify operates. Because when BibClassify finds a match for a composite keyword, it will remove the single components of the composite keyword from the final

result set. The comparison shows correlation coefficients when using the default (non-changed, automatically generated) dictionary, and when the first 50 and 100 most frequent patterns were manually checked and corrected.

Correlation	R1	R2	R3
All patterns	0,666	0,746	0,843
50 most freq kws removed	0,770	0,802	0,872
100 most freq kws removed	0,813	0,811	0,835

We can see that the correlation between BibClassify and R1 is very low for the default, automatically generated dictionary. This poor result is mainly due to the automatically applied stemming. To illustrate this situation, consider the concept of `relativity` theory which in the thesaurus had `relativity` listed as a synonym. The automatically stemmed entries created by Porter stemmer were:

```
relativity → rel
relativity theory → rel the
```

Which resulted in 8950 occurrences extracted, compared to the 90 recognized by BibClassify. This pattern is clearly wrong and matches erroneous entries:

```
relations (1909), relator (8), Relic (20), RELIC (4), rel (54),
related (3), relators (12), relation (3167), RELATIONS (2),
relates the (81), Relation (62), relational (2), Relics (2),
Relative (107), RELATION (9), relation the (2), Related (59),
related theories (1), relatives (7), Relativity (175), RELATED
(2), Relating (2), Relating the (2), related theory (3), relics
(43), RELATIVITY (4), RELATIVE (2), Relate the (1), relative
(2158), relic (705), Relations (44), relate the (37), related the
(6), relating the (52), relations the (6), relativity &
relativity theory (- the rest)
```

After manually checking and fixing the ambiguous, most frequent patterns, results improved considerably as the table shows. We shall note that fixing the first 50 most ambiguous entries had considerable effect. The R2 column shows the effect of POS disambiguation, interestingly the correlation with BibClassify is slightly lower than against the R1/50 run, however, this could be explained by the fluctuation of patterns. When we changed the dictionary, for each run the extraction was different. So in the R1 run we had more deviant patterns in the first 50 entries than in the R2/50 run. And correcting them had more impact.

The R3 results exhibit other interesting behaviour. Firstly, the extraction algorithm was tweaked to more closely resemble BibClassify⁷ which resulted in higher correlation coefficient, in other words SEMAN works more like BibClassify. But results did not continue to improve after more wrong patterns were corrected – this is a rather puzzling result at first. It shows that more accurate patterns had an adverse effect on the correlation,

⁷ As noted before, BibClassify will remove the elements of the composite keyword match from the final result set.

that SEMAN was then picking less keywords, being perhaps less forgiving. But recall that more than 10 thousand patterns are recognized by both system and the most diverging patterns are not always top most frequent ones. In fact, if we were able to sort patterns by the most significant difference between SEMAN and BibClassify, we could achieve 0.90 correlation by fixing the first 64 entries and 0.92 after fixing the first 100 diverging patterns. But since in the normal cases we do not know which patterns differ most, we must simply look at the most frequent matches and pick the wrong extractions from the diagnostic information provided by SEMAN. However, in general SEMAN seems to recognize roughly the same number of patterns. Of course the precision could be greatly improved should we spent more time on tweaking the dictionary. After all, BibClassify contains number of hand-crafted rules that are there to improve the accuracy and filter out false matches.

On average, for each document SEMAN and BibClassify agree on 79.9% of keywords – i.e. out of the all keywords that BibClassify found for each file, 79.9% of them were found in the same file by SEMAN. The missing 20.1% can be attributed to the following:

- BibClassify works with hidden regular expressions, for example D-brane contains also a regular expression `/D[\dp][\s-]branes*/` that matches `D1-brane`, `D2-branes`, `Dp brane` etc. Because SEMANs' dictionary was generated automatically, these patterns are missing. And this example also shows that for some cases, the regular expression pattern is more elegant than having the possible combinations listed in the dictionary; which is often the case for chemical and physical elements.⁸
- Both of the tools pick composite keywords differently, SEMAN searches around the matched keyword, up to certain distance, order is not important. If a composite keyword is found, its components are not available for further lookups (unless they make up part of other composite keywords)
- Finally, for BibClassify tokenization is not so important because it works with the whole plain text, but in the case of SEMAN incorrect tokenization has a big negative impact. For example, if the pattern `K*(892)` is split into 4 tokens, SEMAN must first join them into one token, and certain patterns are simply missed if the tokenization is not correct.

But for the purposes of comparison, we can conclude that the matching mechanism of SEMAN compares well to the regular expression engine matching. But built on the simplified morphological analysis and using the prefix-stem-suffix pattern it is also faster. The difference in speed will increase with the number of patterns. Because for BibClassify usually it is impractical or impossible to decide what pattern should be omitted from matching for a given text.⁹

In SEMAN the matching is done differently (on nodes of a graph), and as it turns out the Information Extraction system inside GATE works the same way. Items are presented as a serialisable stream of tokens which allows for a greater degree of flexibility. Dedicated extraction workflows can be created for special cases while keeping the level of complexity relatively low. And the matching speed is superior to the regular expression matching

⁸ The future version of the system should recognize certain classes of characters and expand them automatically; probably by generating entries automatically from the dictionary prescription.

⁹ And in fact, such optimisation would be a part of the regular expression engine.

mechanism. Due to our ability to effectively apply caching. The following graph shows the impact of such a strategy on the processing of 20 000 documents with more than 1.6 million unique tokens.

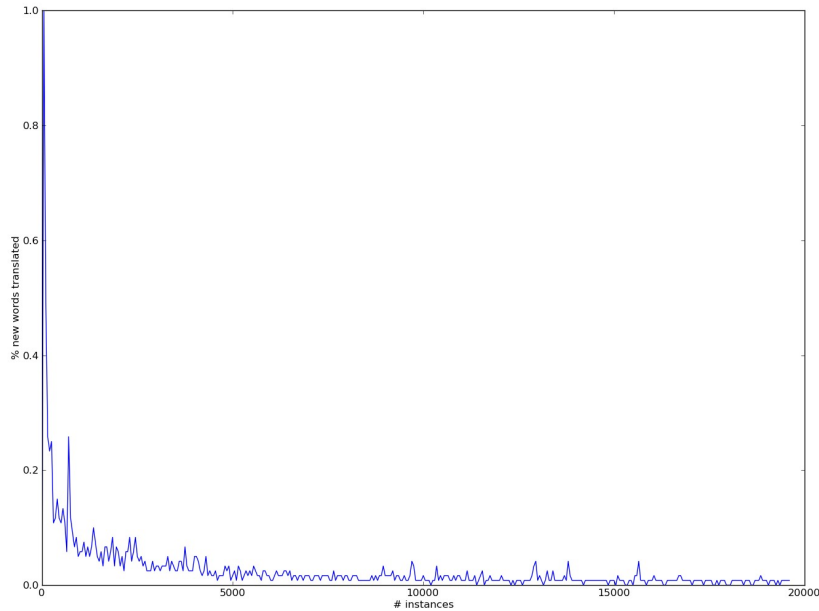


Illustration 25: Percentage of 'yet-unseen matches' discovered as we continue processing a corpus of 20.000 documents

The graph shows the percentage of new tokens as they are discovered during the translation. It is apparent that for the first documents the ratio of new tokens is very high (100%), but quickly diminishes so that after the first 3000 documents, we have already seen more than 95% of all valid patterns. Which shows that after the first few thousand documents, the translation speed can increase considerably.

V.2 SEMANTIC AMBIGUITY

We have seen that the matching mechanism of SEMAN is to a large extent, at least in its results, comparable to the regular expression engine. Even with the automatically derived patterns, out of which only the first 100 most frequent were manually corrected, we were able to arrive at results that correlated closely with much more complete and sophisticated prescriptions of the BibClassify extractor. SEMAN is much faster, especially if the number of dictionary entries grows considerably.

However, more difficult challenges are ahead. Even if we can conclude that the matching mechanism works well – there is the problem of semantic ambiguity. Because it is one thing to be able to extract occurrences of certain patterns from the relatively narrow subject domain corpus, and quite another to select a correct concept out of several possible alternatives, even if we are not interested in understanding the meaning of sentences. But for content analysis to be reliable, we need to produce reasonably accurate data, therefore we have yet to establish how well can SEMAN cope with the problem of ambiguity.

For that purpose, I propose to look at the problem of an automated document classification where we can compare the performance of various combinations of the features, using unprocessed data as well as data that SEMAN offers. For this task we will use two corpora. The previously mentioned corpus of HEP fulltexts for which we possess the subject taxonomy. It will represent the domain specific collection of documents. And another corpus, much broader in its scope, a multidisciplinary collection called 20 Newsgroups.

The 20 Newsgroups data set¹⁰ is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. The 20 newsgroups collection has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g. misc.forsale / soc.religion.christian). Here is a list of the 20 newsgroups, partitioned (more or less) according to subject matter:

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

In our case, the goal of the classification is, based on the features of a document, to choose the group to which the document belongs. We use the supervised learning and the state-of-art classification engine based on support vector machines (SVM). They were shown to achieve the best results in the text categorization tasks (Hsu, C.-C. Chang, and C.-J. Lin 2003; Joachims 1998; “SVM-perf: Support Vector Machine for Multivariate Performance Measures” n.d.) but also because SVM will allow us to work with very large number of features for reasons we will describe later. SVM is a popular technique for classification tasks that involve training and testing data. Each instance in the testing/training set contains one target value (class label) and several attributes (features). The goal of SVM is to produce a model which predicts the target value of the instance after inspection of its attributes. We will be comparing results of the classification based on plain text features against results of the classification based on SEMAN translated features.

¹⁰ <http://people.csail.mit.edu/jrennie/20Newsgroups/>

Given a training set of instance label pairs $(x_i, y_i), i=1, \dots, l$ where $x_i \in R^n$ and $y_i \in \{1, -1\}^l$ the SVM require the solution to the following optimization problem:

$$\begin{aligned} \min(w, b, \xi) \quad & \frac{1}{2} w^T w + C \sum_{i=1}^l \xi_i \\ \text{subject to} \quad & y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned}$$

The training vectors x_i are mapped into a higher (maybe infinite) dimensional space by the function ϕ . Then SVM finds a maximum margin hyperplane that best separates the training instances in this higher dimensional space – i.e. (all) the points of the opposite classes are at the greatest possible distance far away from the separating line. $C > 0$ is the penalty parameter of the error term and SVM works with various kernel functions. These allow the mapping of the the data points into a much higher- and possibly infinite-dimensional inner product space in which it is possible to find a better separating hyperplane. It is thanks to these kernel trick mappings that SVM classification can be used for non-linear classification problems.

Many kernels exist and new ones are constantly researched, but the traditional ones are:

- linear $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (Y x_i^T x_j + r)^d, Y > 0$
- radial basis function (RBF): $K(x_i, x_j) = \exp(-Y \|x_i - x_j\| + r)^d, Y > 0$
- sigmoid: $K(x_i, x_j) = \tanh(Y x_i^T x_j + r)$

Where Y , r , and d are kernel parameters (r is a constant trading off the influence of higher-order versus lower-order terms in the polynomial and sigmoid and the other constants)

But for this task, we will use the SVM implementation coming from LIBLINEAR (Fan, K. W. Chang, Hsieh, Wang, et al. 2008) where the kernel function is replaced with a loss function called L1-SVM.

$$\max(1 - y_i w^T x_i, 0)^2$$

The particular implementation of LIBLINEAR performs much faster than the traditional SVM kernel implementations and is specially suitable for the classification of documents with high number of features. In our case we have to deal with more than 100 thousands of them.

For both corpora, we use the 1/10 split – ie. first a model is built using the training set of 90% documents and the model is evaluated on testing set of 10% documents. The process is repeated 10 times, each time with a different testing and training sets. The results are averaged and we will report the precision, recall and the harmonic measure of the classification for each group.

Usually, the two metrics of precision and recall use two sets of data, one assigned by a human and the other assigned by algorithm. The first set is considered to be “correct” and is often called the “golden standard”. The results of the computation are then compared against this golden standard, precision (P) expresses the number of matching (positive) hits as a proportion hits that were retrieved, and recall (R) is the proportion of the correct hits that exist in the whole collection.

$$P = \frac{\text{number of correct hits}}{\text{number of retrieved hits}}$$

$$R = \frac{\text{number of correct hits}}{\text{number of relevant hits}}$$

The harmonic f-measure we use is then a harmonic mean (sometimes called F1 score)

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

V.2.1 Feature selection and scaling

As is discussed in literature (Forman 2003, n.d.) the selection of features for classification may considerably influence the final results. However, as was shown by Forman as well, these conclusions are not so clear in the area of text categorization. In the experiments conducted by Forman, the different methods of feature selection bore less significant impact on the results of classification using SVM classifier. The selection was important mostly as a way to limit the computational time and memory resources.¹¹ This might stem from the fact that SVM is designed to pick-up the most significant data points (the support vectors) that separate the document classes.

This might be both blessing as well as curse in our case. Blessing, because we can use all the features that we have at our disposal for the classification task without having to select the most significant ones. So we want to compare two classification results that are based on two non-directly comparable data sets. It is perhaps apparent that a wrong feature selection in the case of one dataset could make the results of the second classification seem unjustly better. From this view it seems more appropriate to conduct no feature selection rather than introduce more errors into the comparison.

¹¹ The cost of this approach is acceptable, as classification tasks finish quickly, however memory limitations are still a considerable issue. For the comparison, we were obliged to build a custom framework for feature selection and scaling, as it was not possible to use a more standard and known tools such as Weka (<http://www.cs.waikato.ac.nz/ml/weka/>). While Weka could work with no more than 20 thousand features, filling up 3GB of RAM, our custom framework had to handle at times more than 200 thousand features with much lower memory footprint.

However, the curse of the comparison comes from the way in which SVM works. It is not easy to inspect the SVM model and find out which features had the greatest impact and why (recall that SVM will pick up the hyperplane that maximizes the distance from the data points after they were transformed into higher-dimensional space). The SVM trained model will represent the model trained on many features where only a certain number of these features is significant, but we do not know exactly which of them or what their combinations are. We can only compare the final results of the classification. Small differences will be hard to attribute, but trends and bigger disparities will speak more clearly.

So for the reasons outlined above it was decided that we will compare the results of the same classification task but use different features without filtering them. Nevertheless we still need to prepare them.

An intuitive way of determining document features is by focusing on terms according to their frequency – those that appear most frequently are perhaps more important. However, this view is too simple as certain terms appear too often and provide no additional information. To give more weight to the candidates that are more important, the inverse document frequency (IDF) scale is employed. IDF weights the term according to its frequency in the whole corpus, so that the very frequent terms become less important and the relatively rare terms possess more discriminative value. Then the traditional TF×IDF statistics combines the two measures into a single term metric. Given a candidate token t in a document d , TF×IDF computes the following:

$$TF \times IDF = \frac{freq(t, d)}{\sum_i freq(t_i, d)} \times -\log_2 \frac{n_t}{N}$$

where $freq(t, d)$ is the occurrence count of term t in document d , n_t is the number of documents with token t and N is the total number of documents in the corpus. The first component in this expression is the term frequency, or the normalized frequency of term t in document d . The second is the negative logarithm of the inverse document frequency, which is larger for less frequent tokens.

There exist many similarity measures for vectors, such as the Dice or Jaccard coefficient for binary vectors, however cosine is by far the most popular in information retrieval and also in many other areas. In order to compute the similarity of the vectors, and because cosine for normalized vectors is a dot product of the vectors, we normalize the TF×IDF based vectors. The vector is *normalized* when its unit length according to the Euclidean norm equals one.

$$|\vec{x}| = \sum_{i=1}^n x_i^2 = 1$$

As for the normalized vectors, because cosine is the dot product, we can use the Euclidean distance to measure similarity between the documents in this vector space. (Christopher D. Manning and Schuetze 1999, 301)

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^n (x^i - y^i)^2}$$

This normalization is also important for the SVM classification. For the purpose of classification, we will create a document matrix consisting of vectors with values in the *document space* – by document space we mean all features that are present in the corpus of the documents. The resulting matrix will be sparse and each data point is a real number ($n < I$). But there is a difference in the dimensionality of the resulting matrix. While for the classification task that is based on the plain text, we will have potentially as many features as there are unique tokens in the corpus¹², so the resulting matrix can be very big. Also errors in parsing or neologisms increase the vector space.

On the other side, the document space of the features that are produced by SEMAN is limited by the size of the taxonomy. In the case of SEMAN, by the number of core semes or the compound semes. Thus the vector space is considerably smaller. The question is whether this reduced space is better, or comparable to the vector space produced from the plain text features.

V.2.2 The comparison

The table below and its accompanying graph contains an interesting story of the data processed with the USL. In this case we do not yet look at the final set of results, but we compare pre-liminary results of classification using certain combination of features to see how they influence the overall final score and also the differences inside the categories. This comparison used 0.8/0.2 split, the training set contained 16.000 documents, while the evaluation set hold the remaining 4 thousand documents from the 20 Newsgroup corpus. A cross-evaluation was not conducted (yet), but the table can illustrate the range of challenges.

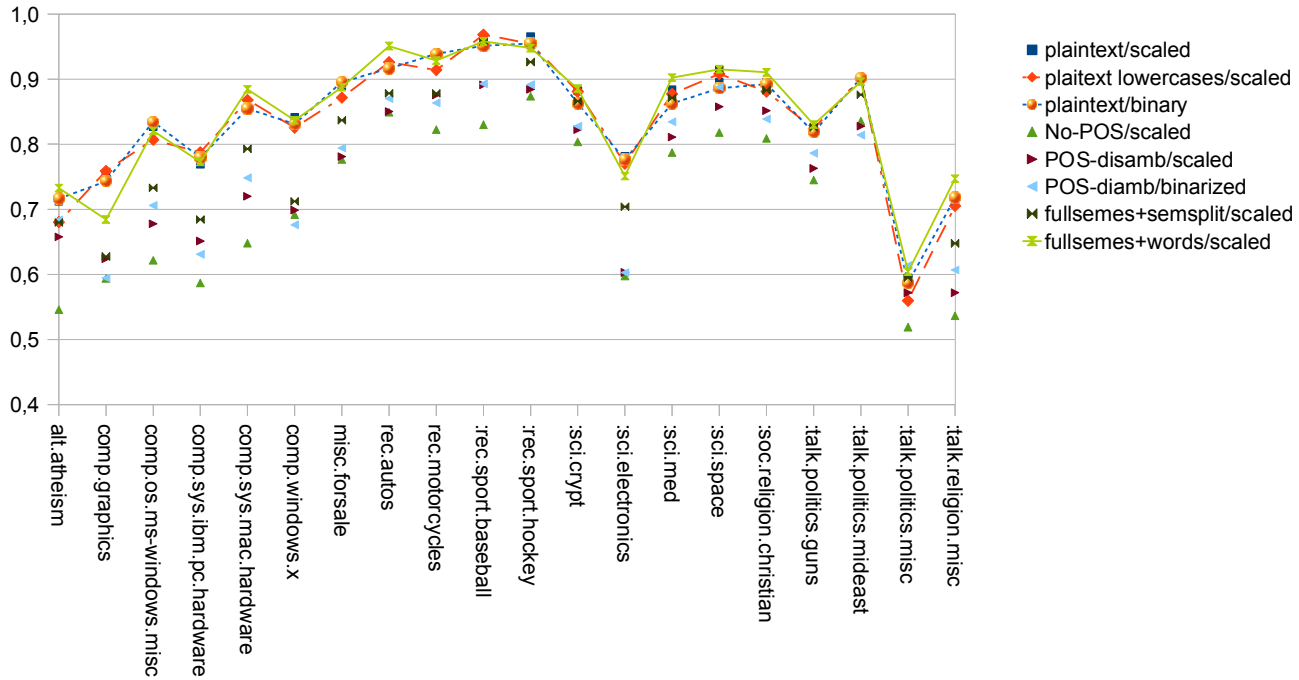
¹² Because we do not group the tokens, even if we potentially could. For example by using latent semantic analysis. But we do not do so intentionally because the goal is to compare results of the analysis using plaintext features against SEMAN translated features.

Group/test	C1	C2	C3	C4	C5	C6	C7	C8
alt.atheism	0.7123	0.6803	0.7162	0.5457	0.6577	0.6839	0.6797	0.7327
comp.graphics	0.7527	0.7590	0.7430	0.5938	0.6242	0.5946	0.6275	0.6843
comp.os.ms-windows.misc	0.8267	0.8069	0.8344	0.6216	0.6776	0.7059	0.7332	0.8202
comp.sys.ibm.pc.hardware	0.7692	0.7874	0.7794	0.5867	0.6513	0.6308	0.6842	0.7727
comp.sys.mac.hardware	0.8522	0.8678	0.8545	0.6479	0.7198	0.7484	0.7929	0.8843
comp.windows.x	0.8412	0.8254	0.8311	0.6918	0.6988	0.6762	0.7123	0.8367
misc.forsale	0.8901	0.8716	0.8954	0.7763	0.7808	0.7940	0.8367	0.8874
rec.autos	0.9200	0.9259	0.9163	0.8489	0.8501	0.8701	0.8780	0.9508
rec.motorcycles	0.9383	0.9140	0.9387	0.8222	0.8757	0.8635	0.8780	0.9284
rec.sport.baseball	0.9562	0.9682	0.9511	0.8300	0.8912	0.8929	0.9574	0.9577
rec.sport.hockey	0.9654	0.9546	0.9546	0.8737	0.8843	0.8914	0.9262	0.9476
sci.crypt	0.8820	0.8808	0.8628	0.8035	0.8224	0.8277	0.8660	0.8856
sci.electronics	0.7815	0.7703	0.7764	0.5976	0.6030	0.6027	0.7039	0.7510
sci.med	0.8835	0.8772	0.8625	0.7870	0.8110	0.8344	0.8710	0.9024
sci.space	0.8950	0.9079	0.8859	0.8179	0.8577	0.8879	0.9139	0.9152
soc.religion.christian	0.8954	0.8808	0.8924	0.8087	0.8515	0.8388	0.8827	0.9103
talk.politics.guns	0.8215	0.8237	0.8182	0.7449	0.7628	0.7861	0.8258	0.8301
talk.politics.mideast	0.9017	0.9017	0.9017	0.8356	0.8277	0.8140	0.8760	0.8960
talk.politics.misc	0.5932	0.5595	0.5870	0.5189	0.5718	0.6138	0.5949	0.6042
talk.religion.misc	0.7115	0.7051	0.7179	0.5364	0.5718	0.6069	0.6477	0.7471
Microaverage								
precision	0.84	0.834	0.837	0.715	0.751	0.759	0.795	0.843
recall	0.84	0.834	0.837	0.715	0.751	0.759	0.795	0.843

The combination of features in this run is the following (it should be understood that whenever the term *scaled features* is used, it is the normalized cosine similarity as described above):

- C1 plaintext: features are roughly the words of the text
- C2 plaintext lowercase: as above, but we limit the number of features by disregarding case differences
- C3 plaintext binary: instead of the cosine similarity, the data points are either ones or zeros (vector length is not equal one)
- C4 No-POS scaled: translation using only semes, automatically selecting the first concept if several translations are possible
- C5 POS-disambiguated scaled: the disambiguation uses POS information
- C6 POS-disambiguated binary: as previous but data points are simply zeros or ones
- C7 fullsemes + semsplit/ scaled: the features are made of composite semes as well as individual semes (counting both the aggregate and core terms)

- C8 Fullsemes + words / scaled: the features include semes and where seme is not available, then words



The first three columns of the table show results of the multi-label classification using only the plaintext features – any token which is present in the document will be used by the SVM for the classification. The first column displays results of the classification using the frequency count of the token scaled by the cosine. The second column shows an interesting variation in the results, when fulltext features were normalized, using only the lowercase form, the number of features decreased considerably – we do not draw any conclusions here, because the set was not cross-validated – but it is interesting to note the negative effect on the accuracy of the final results. It is true that in four categories we achieved best results using the lowercased normalized form, but the difference did not seem large and after this initial run, it was decided to use plaintext without normalization. Overall, the first set fared better by only 0.12%.

The third column represents another interesting variation to the first row-set. This time the features are not normalized, neither scaled, but the presence of a feature is simply marked. As we can see, the scaled feature classification is generally more accurate than the binary features, yet the differences (in this test run) were not dramatic – only 0.07% – thus even if we decided to use the most accurate method for a comparison with the classification on semes, it should be noted that the binary features represent the interesting variant which is computationally easier as no cosine scaling is needed. This finding is in line with the results reported in (Hopkins and King 2010; King, Knowles, and Melendez 2010) where

also the presence or absence of a feature is counted without measuring the relative importance – the weight of the feature. Though it might seem counterintuitive, the experimental results repeatedly show good performance.

All the remaining columns then represent results somehow influenced by the process of translation. The fourth column (C4) shows classification when only semes are used – ie. the text is processed, the appropriate matching tokens and groups of tokens are identified, translated into semes and only the semes are used for classification. But we did not attempt to select the appropriate translation for a pattern that matched the token. Because the WordNet lists several definitions of the term, and the system is not able to decide on the most appropriate one, we use all of them. Such an approach results in added ambiguity and the SVM machine is in fact incapable of finding the most distinctive feature vectors for classification. Clearly, this method is the simplest and it does produce the worst possible results. 18 out of 20 groups achieved the lowest accuracy, overall only 71.50%.

However this result is very interesting on its own. We can observe the effect of introducing more ambiguity into the data (by inclusion of all possible translations). And we can see the change from 'the-worst procedure', towards improvements when some sort of disambiguation was employed.

The next column presents the results after the part-of-speech disambiguation. In particular, we are using only nouns, adjectives and adverbs for the purpose of classification. The number of definitions (semantic composites used) decreased by more than 20% and with it also the artificially induced ambiguity. The SVM is able to identify support vectors better and we have reached the 75% accuracy. As is discussed in (Schrodt 2009; Schrodt and Gerner n.d.) this level of accuracy is acceptable for certain content analysis applications. When we think of the fact that no special processing (besides the POS tagging, which SEMAN incorporates) was needed, this option might seem attractive. Especially if we recall that only a very limited set of tokens is used for classification, the number of features is lower by more than 50%. On the other hand, the comparison with the previous column also shows that without incorporating the NLP technology, the usefulness of SEMAN decreases rather significantly.

It is interesting to observe that the cosine scaling has an effect on the final accuracy, but at the same time the simple presence or absence of a sem (feature) is a good indicator and SVM is able to use it for the purpose of classification. In this case it meant an increase of overall precision by 0.8%. This is not surprising because the features are selected from the limited set of pre-defined words. Thus presence or absence of a feature in this narrower space may be a somewhat stronger indicator of a category than in the case of plain text features, which are potentially much more numerous than semantic features.

As we can see classification based only on the semantic features obtained the worse results if no disambiguation was employed, and improved by 9% to almost the magical range of 80% when a more careful sense disambiguation was employed. I was curious whether a more complete sense disambiguation could improve the results even further. To test this, I have used the word sense disambiguation (WSD) software by Pedersen (Pedersen and Kolhatkar 2009). While many variants of the WSD exist and the Pedersen's solution is not the most accurate, it achieves substantial F-Score of 76.02% over general domain corpus and the algorithm itself finished as the second best in the SemEval07¹³ competition. (Kolhatkar 2009) But while not being state-of-art, still we can reasonably expect that it

13 <http://nlp.cs.swarthmore.edu/semeval/tasks/index.php>

does a much better job than a simple disambiguation based solely on POS information. Unfortunately, the cost of running the WSD on a relatively small corpus of 20 thousand documents is very high. A small cluster of eight machines was working for 32 hours just on the disambiguation task. So this option is not viable for the real life tasks, but it may provide interesting comparison.

	Microaverage		Macroaverage	
	Precision	Recall	Precision	Recall
Semes	0,8174	0,8174	0,8199	0,8174
Semes*	0,8070	0,8070	0,8070	0,8070

The table above summarizes the results of classification after the WSD method was employed. We are using only the seme features in the classification and indeed the results improved by almost 2%. The second row, marked as “Semes*”, shows an interesting variation to the processing when the last element of the concept definition was removed. For example the last element from the definition of the concept car as in the example below :

```

car = entity
    physical_entity
        object
            whole
                artifact
                    instrumentality
                        container
                            wheeled_vehicle
                                self-propelled_vehicle
                                    motor_vehicle
                                        car

```

If we remove the last element (car) the definitions are thus more general, more objects are clustered together and this results in decreased accuracy. This is valuable observation for the task of document classification but in the case of content analysis studies, we often want the details to be suppressed. In this case, we continued using the unabridged definitions that gave us 2% increase in the accuracy. Yet the cost of this improvement is indeed high, and the procedure of WSD makes whole procedure too slow. It is obvious we cannot use it in the real-life application. But it is good to know that there is a space for improvement in better disambiguation and the direction clearly goes into more precise assignment of meaning. But it is questionable whether we could achieve bigger gains with better and smarter disambiguation algorithms. To see why, we have to look more closely at the corpus we are translating.

```

Number of files: 19618
Total number of tokens: 5382275
Number of translated tokens: 2050193 (38.1%)
Multi token expressions recognized: 33868
Average number of tokens per doc: 274.35
Number of tokens translated/doc: 104.51

```

From the statistics above we can see that there are almost 5.4 million tokens but only 38% of them recognized and translated into the semantic codes. The average number of tokens per document is also rather low because the corpus is made of newsgroup messages that tend to be shorter and made of reactions on some previous replies.

So if we use only the semes for classification, it is quite possible that many messages contain very little information. It is actually surprising that we were able to achieve almost 80% accuracy given such ratio of translation (or 82% with the more elaborate word sense disambiguated corpus). However, we may be close to the limits of accuracy and further improvements may not be easy to obtain, certainly not if we do not have enough data for every document or when the translation rate is low. And that will often be the case with content analysis studies in which the research is focused on a pre-defined variables omitting all the rest.

Nevertheless, our test is still valid. The preliminary results show that the accuracy of classification improves when we employ the POS disambiguation techniques, and as expected, it improves even further if word sense disambiguation is applied.

Because WSD is prohibitively expensive, we will not use it for cross-evaluation. Instead our focus is directed towards comparison of results from three datasets. We will compare the classification that uses scaled plaintext features (a baseline for comparison) against the classification that is using only semes, and finally the classification that is based on the combination of semes and plaintext features. And we will report results for both corpora, the High Energy Physics papers and the 20 Newsgroups. All the numbers reported below are computed with 1/9 split, cross-validated across the whole corpus. Discussion follows the results.

V.2.2.1 HEP corpus

Classification that used plain text features only:

```
=====
class    accuracy    precision    recall    fscore
=====
1.0    0.968306010929    0.915851831624    0.896167702274    0.899704228865
2.0    0.945901639344    0.845513485885    0.832473118280    0.833114882782
3.0    0.967213114754    0.912607085972    0.905125653782    0.905881037602
4.0    0.896721311475    0.69195076879    0.637663416285    0.647404952567
5.0    0.910928961749    0.716733180612    0.730740651799    0.727272378357
6.0    0.955737704918    0.839964727110    0.911623775564    0.896046606180
=====
microaverage:
precision: 0.822404
recall: 0.822404

macroaverage:
precision: 0.820439
recall: 0.818967

confusion matrix:
=====
.    1.0    2.0    3.0    4.0    5.0    6.0
=====
1.0    55.4    0.8    0.0    1.8    3.2    0.6
```

2.0	0.6	51.0	0.0	9.2	0.2	0.2
3.0	0.2	0.8	59.2	2.6	1.6	1.0
4.0	2.8	7.8	3.6	37.0	6.4	0.4
5.0	1.6	0.2	2.2	3.2	42.6	8.6
6.0	0.0	0.0	0.0	0.0	5.4	55.8

Classification that used semes only:

class	accuracy	precision	recall	fscore
1.0	0.965217391304	0.890801257967	0.907992918985	0.904209667538
2.0	0.948405797101	0.841617947930	0.862514483169	0.856107023853
3.0	0.962318840580	0.880873873029	0.911067278798	0.904569552113
4.0	0.889275362319	0.684631751227	0.569027161662	0.588327339490
5.0	0.899710144928	0.689191841962	0.673469152597	0.675093119523
6.0	0.939710144928	0.795411355897	0.863831782196	0.848283214879

microaverage:

precision: 0.802319

recall: 0.802319

macroaverage:

precision: 0.797090

recall: 0.797985

confusion matrix:

.	1.0	2.0	3.0	4.0	5.0	6.0
1.0	53.2	0.6	0.0	1.8	1.6	1.4
2.0	0.8	50.4	0.6	6.0	0.4	0.2
3.0	0.2	1.0	55.8	2.6	0.6	1.0
4.0	3.0	8.0	4.4	31.2	7.2	1.0
5.0	2.2	0.2	2.2	4.0	36.8	9.4
6.0	0.4	0.0	0.4	0.2	6.8	49.4

And finally classification based on semes or plain text features (when semes are not available):

class	accuracy	precision	recall	fscore
1.0	0.972173913043	0.923059185503	0.913805996053	0.915147664359
2.0	0.949565217391	0.862409998182	0.836738624762	0.839823333183
3.0	0.966376811594	0.901411032396	0.911590204747	0.908566356542
4.0	0.901449275362	0.707159307258	0.657353101608	0.666329170249
5.0	0.915362318841	0.733111473193	0.744959621275	0.741576664152
6.0	0.957681159420	0.853799936023	0.905574232590	0.893842915910

microaverage:

precision: 0.831304

recall: 0.831304

macroaverage:

precision: 0.830160

recall: 0.828339

confusion matrix:

.	1.0	2.0	3.0	4.0	5.0	6.0
1.0	53.2	0.6	0.0	1.4	2.4	0.6
2.0	0.6	48.6	0.2	8.2	0.2	0.2
3.0	0.2	1.0	56.0	2.8	0.8	0.6
4.0	2.4	6.2	3.4	36.0	6.4	0.4
5.0	1.4	0.2	2.4	2.8	41.2	7.4
6.0	0.0	0.0	0.2	0.0	5.2	51.8

And the aggregate view on the comparison of the three sets using the f-score:

Category	Text	Sem	SemText
astro-ph	0.8997	0.9042	0.9151
hep-ex	0.8331	0.8561	0.8425
hep-lat	0.9059	0.9046	0.9099
hep-ph	0.6474	0.5883	0.6701
hep-th	0.7273	0.6751	0.7489
math-ph	0.8960	0.8483	0.8956
Microaverage			
Precision	0.8224	0.8023	0.8342
Recall	0.8224	0.8023	0.8342
Macroaverage			
Precision	0.8204	0.7971	0.8326
Recall	0.8190	0.7980	0.8310

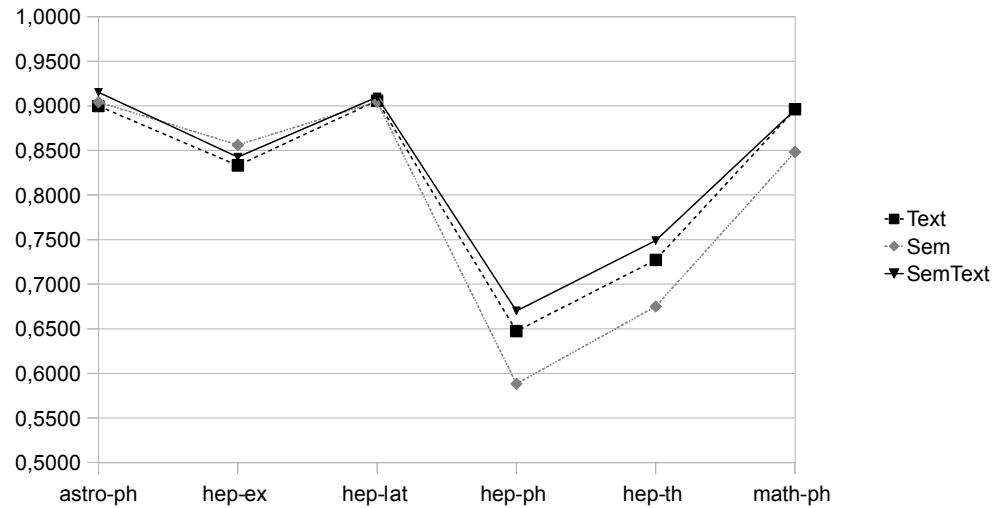


Illustration 26: Comparison of the classification results that were based on different datasets

V.2.2.2 The 20 newsgroups corpus

Due to formatting issues, we will present the aggregate views first.

Category	Text	Sem	SemText
alt.atheism	0.8759	0.8702	0.9135
comp.graphics	0.8092	0.7774	0.8466
comp.os.ms-windows.misc	0.8299	0.7512	0.8588
comp.sys.ibm.pc.hardware	0.7975	0.7239	0.8096
comp.sys.mac.hardware	0.8638	0.8005	0.8949
comp.windows.x	0.8837	0.8108	0.9055
misc.forsale	0.8620	0.8367	0.8812
rec.autos	0.9004	0.8834	0.9280
rec.motorcycles	0.9389	0.9166	0.9436
rec.sport.baseball	0.9475	0.9494	0.9733
rec.sport.hockey	0.9581	0.9562	0.9726
sci.crypt	0.9160	0.9395	0.9531
sci.electronics	0.8230	0.7948	0.8604
sci.med	0.9034	0.9061	0.9244
sci.space	0.9178	0.9189	0.9424
soc.religion.christian	0.9092	0.8769	0.9255
talk.politics.guns	0.9061	0.9120	0.9414
talk.politics.mideast	0.9490	0.9404	0.9630
talk.politics.misc	0.8657	0.8576	0.8981
talk.religion.misc	0.8406	0.8428	0.8712
Microaverage			
Precision	0.8851	0.8851	0.9105
Recall	0.8851	0.8851	0.9105
F-score	0.8851	0.8851	0.9105
Macroaverage			
Precision	0.8868	0.8658	0.9122
Recall	0.8851	0.8633	0.9105
F-score	0.8860	0.8646	0.9114

Illustration 27: F-measure for the document classification of 20 Newsgroups

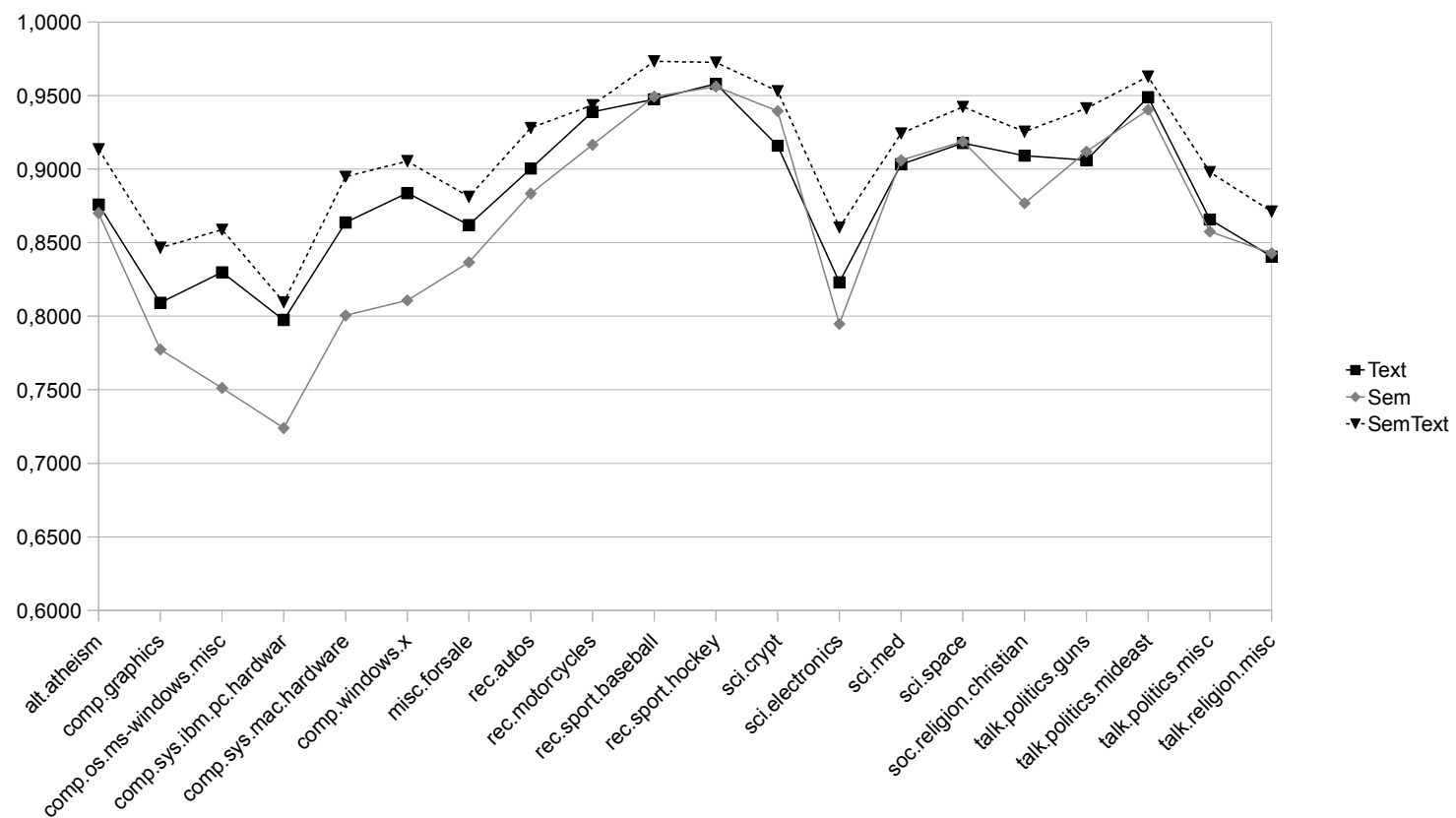


Illustration 28: Aggregate view on the document classification for 20 Newsgroups corpus

Details by class:

class	accuracy	precision	recall	fscore
1	0.988955823293	0.907143805719	0.868646464646	0.875901195562
2	0.979267068273	0.779618322021	0.81723891981	0.809233712835
3	0.982630522088	0.824265772538	0.832494949495	0.829899874466
4	0.979016064257	0.785226103624	0.801414141414	0.797484344986
5	0.986495983936	0.867195033015	0.863199340342	0.863773402962
6	0.988855421687	0.894637648971	0.881444444444	0.883744612662
7	0.98483935743	0.832618362789	0.869864151721	0.862036992037
8	0.989859437751	0.89746234518	0.901626262626	0.90043989812
9	0.994779116466	0.960769483975	0.933727272727	0.93893180529
10	0.99437751004	0.939237436237	0.949808080808	0.947535238122
11	0.995532128514	0.951751381549	0.959878787879	0.958141683194
12	0.992520080321	0.938077983863	0.910858585859	0.915968944275
13	0.982831325301	0.836865064337	0.820464646465	0.823005425113
14	0.990813253012	0.915764882371	0.900717171717	0.903363536359
15	0.992570281124	0.936015480691	0.913555555556	0.917819804719
16	0.988403614458	0.85994008218	0.923563182849	0.909160846994
17	0.989307228916	0.878568680392	0.913585858586	0.906071614625
18	0.994979919679	0.951446718494	0.948747474747	0.949029829853
19	0.98765060241	0.888879275279	0.860414141414	0.865722361043
20	0.986596385542	0.890991105369	0.831141414141	0.840552102942

Classification that used plain text features only:

microaverage:
precision: 0.885141
recall: 0.885141

macroaverage:
precision: 0.886824
recall: 0.885120
confusion matrix:

.	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0	20.0
1	86.6	0.6	0.2	0.0	0.0	0.0	0.1	0.2	0.1	0.1	0.0	0.2	0.1	0.7	0.6	3.9	0.3	0.7	1.4	3.9
2	0.2	81.4	4.2	3.1	1.5	3.8	1.3	0.3	0.2	0.2	0.2	0.5	1.0	0.2	0.5	0.0	0.4	0.1	0.3	0.2
3	0.0	3.2	83.0	5.5	2.2	2.2	0.7	0.1	0.0	0.0	0.1	0.1	1.0	0.1	0.3	0.3	0.3	0.0	0.2	0.4
4	0.0	3.6	5.1	79.9	3.7	1.0	2.3	0.5	0.1	0.2	0.0	0.2	2.6	0.3	0.1	0.0	0.1	0.0	0.0	0.0
5	0.2	1.4	2.6	3.2	85.8	0.3	2.6	0.2	0.1	0.2	0.1	0.3	1.7	0.2	0.0	0.1	0.0	0.1	0.1	0.2
6	0.1	5.4	2.2	1.2	0.3	87.8	0.5	0.0	0.0	0.2	0.0	0.1	0.8	0.3	0.2	0.2	0.1	0.0	0.1	0.1
7	0.1	0.4	1.0	2.1	1.7	0.1	86.2	2.1	0.7	0.1	0.3	0.3	2.1	0.3	0.5	0.5	0.3	0.0	0.1	0.2
8	0.0	0.3	0.2	0.7	0.1	0.2	2.0	89.9	1.5	0.3	0.2	0.0	2.3	0.7	0.1	0.2	0.6	0.0	0.4	0.0
9	0.0	0.3	0.1	0.0	0.1	0.1	1.8	2.1	93.0	0.2	0.0	0.1	0.4	0.3	0.1	0.2	0.2	0.2	0.2	0.2
10	0.2	0.3	0.0	0.2	0.1	0.3	0.6	0.0	0.0	94.7	2.3	0.0	0.1	0.1	0.0	0.2	0.2	0.0	0.3	0.1
11	0.1	0.4	0.0	0.1	0.2	0.1	0.2	0.1	0.2	1.7	95.7	0.1	0.1	0.0	0.0	0.0	0.2	0.1	0.2	0.2
12	0.3	1.6	0.7	0.1	0.2	0.5	0.3	0.1	0.0	0.3	0.2	90.8	1.4	0.3	0.2	0.2	1.4	0.0	0.9	0.2
13	0.3	1.9	0.6	4.1	2.0	0.4	2.1	2.3	0.1	0.3	0.1	1.4	81.8	0.9	1.0	0.2	0.0	0.0	0.1	0.1
14	0.4	1.2	0.0	0.6	0.5	0.4	0.5	0.7	0.2	0.7	0.3	0.2	1.2	89.8	1.0	0.2	0.7	0.3	0.7	0.1
15	0.7	1.2	0.2	0.3	0.2	0.3	1.0	0.1	0.0	0.3	0.2	0.5	0.8	1.3	91.0	0.4	0.4	0.0	0.2	0.5
16	1.8	0.3	0.2	0.3	0.0	0.2	0.4	0.2	0.0	0.4	0.1	0.0	0.0	0.7	0.4	91.9	0.3	0.6	0.4	1.3
17	0.2	0.3	0.3	0.1	0.2	0.2	0.3	0.7	0.2	0.4	0.1	0.7	0.3	0.0	0.1	0.0	91.0	0.6	3.2	0.7
18	0.6	0.2	0.0	0.1	0.2	0.1	0.2	0.3	0.3	0.4	0.1	0.4	0.0	0.2	0.1	0.8	0.2	94.4	0.8	0.1
19	1.1	0.2	0.1	0.1	0.0	0.0	0.2	0.3	0.1	0.1	0.3	0.9	0.2	1.0	0.5	0.8	5.1	1.5	85.7	1.4
20	2.6	0.3	0.2	0.2	0.1	0.2	0.2	0.1	0.0	0.1	0.3	0.0	0.2	0.8	0.5	7.3	1.9	0.7	1.1	82.8

class	accuracy	precision	recall	fscore
1	0.987957852484	0.891908338063	0.865585858586	0.870165673917
2	0.976869041646	0.763108085306	0.781414141414	0.77735137871
3	0.973858504767	0.732415428331	0.757121212121	0.751204542144
4	0.970697441044	0.698504233401	0.731242424242	0.723943206466
5	0.979277471149	0.788779684158	0.804252525253	0.800545438945
6	0.981886603111	0.825788471101	0.807599876314	0.810798707844
7	0.983542398394	0.834713030879	0.837636363636	0.836683554174
8	0.988058203713	0.879612723121	0.884686868687	0.88341682283
9	0.992373306573	0.934516742996	0.912707070707	0.916594015773
10	0.994731560462	0.94566682551	0.950868686869	0.94940923324
11	0.99633718013	0.974410036225	0.951898989899	0.956172146464
12	0.994430506774	0.951308976218	0.936797979798	0.939519358921
13	0.980180632213	0.810708573362	0.791434343434	0.794752904751
14	0.991419969895	0.925273157198	0.901747474747	0.906090900732
15	0.992674360261	0.937292041746	0.914606060606	0.918854298535
16	0.98554942298	0.835876688779	0.888545454545	0.876895252628
17	0.98986452584	0.884524563169	0.919676767677	0.912004509976
18	0.994330155544	0.948104707828	0.938848484848	0.940409482414
19	0.986452584044	0.872599403025	0.854383838384	0.857622208471
20	0.986151530356	0.881335372379	0.834800659658	0.842798287485

Classification that used semes only:

microaverage:
precision: 0.863322
recall: 0.863322

macroaverage:
precision: 0.865823
recall: 0.863293
confusion matrix:

.	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0	20.0
1	86.3	0.0	0.0	0.1	0.2	0.0	0.0	0.3	0.0	0.1	0.1	0.3	0.1	0.4	0.4	5.0	0.2	1.2	1.4	3.6
2	0.1	77.9	5.7	2.2	2.7	5.2	1.0	0.2	0.1	0.5	0.1	0.6	1.7	0.4	0.7	0.2	0.0	0.0	0.1	0.3
3	0.0	5.3	75.5	7.2	2.7	4.2	0.6	0.3	1.0	0.2	0.0	0.0	1.2	0.2	0.6	0.0	0.1	0.0	0.4	0.2
4	0.0	2.6	8.4	72.9	6.6	1.2	2.7	0.4	0.0	0.1	0.1	0.3	3.4	0.3	0.1	0.1	0.3	0.0	0.0	0.2
5	0.0	1.7	1.7	8.1	80.1	1.1	3.5	0.3	0.1	0.0	0.0	0.1	2.2	0.2	0.1	0.0	0.0	0.0	0.3	0.1
6	0.1	5.8	6.6	1.8	0.7	80.2	0.7	0.0	0.3	0.1	0.1	0.3	1.2	0.2	0.5	0.1	0.1	0.1	0.1	0.3
7	0.3	0.8	1.5	2.6	2.1	0.3	83.5	2.6	0.7	0.6	0.1	0.3	2.6	0.4	0.5	0.1	0.2	0.1	0.3	0.1
8	0.0	0.4	0.6	1.0	1.2	0.3	2.1	88.2	1.9	0.4	0.0	0.1	1.5	0.1	0.2	0.2	0.5	0.2	0.5	0.3
9	0.1	0.1	0.1	0.2	0.4	0.3	1.5	3.2	91.0	0.4	0.1	0.0	0.7	0.4	0.4	0.1	0.5	0.0	0.1	0.1
10	0.1	0.3	0.1	0.2	0.2	0.3	0.2	0.5	0.1	94.8	1.1	0.0	0.3	0.4	0.0	0.2	0.2	0.0	0.4	0.3
11	0.1	0.2	0.2	0.3	0.0	0.4	0.5	0.0	0.5	1.5	94.9	0.0	0.1	0.0	0.2	0.0	0.3	0.1	0.3	0.1
12	0.1	0.7	0.5	0.7	0.2	0.9	0.1	0.0	0.1	0.1	0.0	93.4	1.3	0.1	0.0	0.2	0.7	0.1	0.4	0.1
13	0.1	2.5	1.1	5.0	2.3	1.1	1.9	2.8	0.2	0.4	0.2	1.1	78.9	0.8	0.4	0.4	0.1	0.1	0.1	0.2
14	0.5	0.9	0.3	0.6	0.8	0.6	0.3	0.7	0.4	0.6	0.2	0.1	1.1	89.9	0.6	0.3	0.9	0.0	0.8	0.1
15	0.9	1.6	0.4	0.5	0.5	0.4	0.5	0.1	0.2	0.1	0.0	0.2	0.5	1.1	91.1	0.4	0.2	0.1	0.7	0.1
16	3.3	0.5	0.2	0.4	0.4	0.2	0.2	0.1	0.3	0.0	0.1	0.1	0.4	0.5	0.0	88.5	0.3	0.6	0.8	2.7
17	0.4	0.0	0.0	0.1	0.3	0.1	0.2	0.2	0.3	0.1	0.0	0.7	0.1	0.3	0.2	0.2	91.7	0.4	3.9	0.5
18	1.1	0.2	0.1	0.2	0.2	0.0	0.2	0.1	0.1	0.2	0.2	0.2	0.1	0.3	0.2	0.8	0.2	93.6	1.1	0.6
19	0.9	0.3	0.2	0.2	0.2	0.2	0.4	0.5	0.2	0.1	0.0	0.4	0.0	1.1	0.7	0.9	5.5	1.4	85.1	1.3
20	2.5	0.4	0.2	0.2	0.1	0.2	0.0	0.0	0.0	0.1	0.1	0.0	0.2	0.1	0.3	8.5	1.9	0.8	0.8	83.1

class	accuracy	precision	recall	fscore
1	0.992222779729	0.934549948872	0.908757575758	0.91350404526
2	0.982890115404	0.814030571136	0.855575757576	0.846565976643
3	0.985047666834	0.843254871919	0.863595959596	0.858802407148
4	0.981083793276	0.813152592256	0.809454545455	0.809588922439
5	0.989563472153	0.897713835825	0.894646464646	0.894854587424
6	0.99071751129	0.9097833921	0.904757575758	0.905490173943
7	0.986803813347	0.853391973073	0.888636363636	0.881173872231
8	0.992574009032	0.923126281502	0.929747474747	0.928036669699
9	0.995183140993	0.964022417812	0.938838383838	0.943635635355
10	0.99708981435	0.967694812191	0.974909090909	0.973329862702
11	0.997691921726	0.984075931178	0.969898989899	0.972574507215
12	0.995885599599	0.967591608244	0.949828282828	0.953106817668
13	0.986051179127	0.861198122052	0.860636363636	0.86043596784
14	0.993226292022	0.943870785497	0.919797979798	0.924361772091
15	0.994731560462	0.954137546197	0.939645846217	0.94237404204
16	0.990416457602	0.882396993256	0.937747474747	0.925460525671
17	0.993125940793	0.918299773179	0.947676767677	0.941368720087
18	0.99633718013	0.964973764457	0.962848484848	0.962988940513
19	0.990868038133	0.923978480887	0.892363636364	0.898091807593
20	0.989563472153	0.923563213983	0.860840032983	0.87122847588

Classification based on semes or plain text features (when semes are not available):

microaverage:
precision: 0.910537
recall: 0.910537

macroaverage:
precision: 0.912241
recall: 0.910511

confusion matrix:

	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0	15.0	16.0	17.0	18.0	19.0	20.0
1	90.6	0.1	0.0	0.1	0.0	0.0	0.3	0.1	0.0	0.0	0.0	0.1	0.1	0.2	0.3	4.0	0.0	0.4	0.6	2.8
2	0.0	85.3	3.9	2.2	1.2	3.3	1.0	0.0	0.2	0.2	0.1	0.3	0.6	0.4	0.7	0.1	0.0	0.0	0.1	0.1
3	0.0	3.8	86.1	4.7	1.2	2.0	0.8	0.0	0.0	0.0	0.0	0.0	0.6	0.1	0.2	0.0	0.1	0.0	0.0	0.1
4	0.0	2.7	6.1	80.7	2.9	1.1	2.4	0.1	0.0	0.1	0.1	0.2	2.7	0.2	0.1	0.1	0.1	0.0	0.0	0.1
5	0.1	1.7	1.0	2.4	89.2	0.5	2.6	0.1	0.1	0.0	0.0	0.0	1.5	0.1	0.0	0.1	0.1	0.0	0.2	0.0
6	0.0	4.1	2.4	0.9	0.1	90.2	0.5	0.1	0.2	0.1	0.0	0.0	0.4	0.1	0.2	0.0	0.1	0.1	0.0	0.2
7	0.1	0.7	0.6	2.5	1.2	0.1	88.5	2.0	0.4	0.2	0.0	0.1	2.1	0.2	0.4	0.1	0.2	0.0	0.0	0.2
8	0.0	0.2	0.0	0.6	0.2	0.3	1.9	92.7	1.2	0.1	0.2	0.0	1.5	0.1	0.0	0.1	0.2	0.2	0.2	0.0
9	0.0	0.1	0.1	0.1	0.1	0.2	1.4	2.4	93.6	0.2	0.1	0.1	0.4	0.5	0.1	0.0	0.1	0.1	0.1	0.0
10	0.0	0.1	0.0	0.1	0.2	0.2	0.3	0.1	0.0	97.2	0.8	0.0	0.2	0.0	0.1	0.2	0.1	0.0	0.0	0.1
11	0.0	0.3	0.0	0.0	0.1	0.2	0.2	0.1	0.1	1.1	96.7	0.0	0.1	0.2	0.0	0.0	0.2	0.1	0.1	0.2
12	0.3	0.7	0.6	0.5	0.0	0.4	0.2	0.0	0.1	0.0	0.0	94.7	1.1	0.2	0.0	0.1	0.6	0.0	0.2	0.0
13	0.1	1.2	0.4	3.8	1.6	0.1	2.2	1.4	0.1	0.3	0.0	1.1	85.8	0.7	0.5	0.2	0.1	0.0	0.0	0.1
14	0.3	0.9	0.4	0.3	0.5	0.3	0.3	0.5	0.3	0.4	0.2	0.0	1.2	91.7	0.6	0.2	0.6	0.0	0.8	0.2
15	0.5	1.7	0.2	0.2	0.3	0.1	0.4	0.1	0.0	0.1	0.0	0.1	0.5	0.8	93.5	0.1	0.4	0.0	0.5	0.0
16	1.6	0.5	0.1	0.1	0.0	0.1	0.2	0.2	0.3	0.0	0.1	0.1	0.2	0.3	0.1	93.4	0.2	0.4	0.7	1.0
17	0.0	0.1	0.1	0.0	0.1	0.0	0.4	0.0	0.2	0.1	0.0	0.5	0.2	0.1	0.0	0.1	94.4	0.2	2.6	0.5
18	0.3	0.3	0.1	0.0	0.3	0.0	0.1	0.2	0.1	0.2	0.0	0.3	0.0	0.1	0.1	0.5	0.2	95.9	0.6	0.3
19	0.9	0.1	0.1	0.1	0.1	0.0	0.0	0.4	0.1	0.1	0.0	0.3	0.2	0.7	0.7	0.4	4.0	1.4	88.8	1.1
20	2.2	0.4	0.1	0.1	0.2	0.1	0.0	0.0	0.1	0.1	0.0	0.0	0.3	0.5	0.4	6.6	1.2	0.7	0.8	85.7

We can notice that, as in the first corpus, results based purely on the semes are less accurate than the plain text features. What is positive though is that the difference is often only 2% and we are using only 30% of text features. What is even more positive is the fact that the combined dataset made of semes and plain text features (when semes are not available) again outperforms the baseline classification. From this we can conclude that the translation of words into the semantic codes is relatively accurate even if only POS information is used. This result cannot be stressed enough. It means that without a complicated and expensive disambiguation procedures, we can use the standard, state-of-the-art POS tagging and achieve actually a better performance than the purely text based classification. This is perhaps not so important in the context of the document classification, but it means a lot for content analysis application. Based on this results, we can expect that even in a multi-domain corpus, SEMAN will be able to code the tokens with a reasonable level of accuracy.

V.2.3 Search for significant combinations

So far we were able to establish that the pattern matching mechanism is faster than the regular expression matching mechanism but its results comparable. The automated document classification showed that results depend on the quality of translation and the disambiguation mechanism. But we were able to obtain for our application acceptable level of accuracy using relatively simple but available POS disambiguation. This compensated for the ambiguity of multiple concepts and in fact, the classification that combined the translation with the plaintext features was always superior to the classification based on plain text tokens, even if significantly only for the second corpus. Nevertheless, we could see that the previous stages gave us relatively accurate results and the process of token translation, if prepared carefully, did not introduce unacceptable levels of noise into source data.

The last piece of missing information is about our ability (or inability for that matter) to discover the interesting combination of concepts in the text that was enriched by thousands of semantic codes. By 'interesting' we mean the combinations of semantic codes that are significantly different from the values we would expected in normal situations¹⁴. We shall use the collocation discovery mechanism that was described in the previous chapters and find out whether it also applies to the translated corpora. Recall that the corpus was considerably changed by the inclusion of multiple concept codes. They are not strictly linear as the normal text and they alone account for additional millions of combinations in the case of large collections. Having said that, only a few of the combinations will be interesting to us so it is indeed a search for a needle in the haystack.

Since it would be difficult to evaluate the collocation search against corpora that are usually used in the collocation search (and which do not contain semantic information), we will prepare two synthetic benchmarks – first using a randomly generated corpus, and the

¹⁴ And normal, to continue, is in turn anything that we want to use as a basis for finding the 'strange', 'unnormal' distribution.

second one based on a real translated corpus. Both of the tests should look at how well the collocation discovery mechanism works. We will evaluate the results then using the standard metrics of precision of recall.

In the case of our tests, first a dictionary of an imaginary language is created (for this task it had the vocabulary size of 100.000) and the distribution of the language units follows the Zipf law. Using the dictionary, we are creating a corpus of 20000 documents, each containing a randomly selected number of sentences with 5 to 20 words per sentence. The documents are relatively short, spanning the range of 50 to 400 words per message. Words are selected randomly which results in a relatively very high number of unique collocations – more than it would have been in the real language corpus where certain combinations just do not happen.

The taxonomy used for the purpose of translation is also generated automatically. It is built from 3000 core semes that are randomly combined to build a complete dictionary of 7 000 compound semes; thus our taxonomy contains 10 000 unique concepts. These are assigned in a completely random fashion to the tokens of the dictionary until the ratio of translated words is covered. In this experiment we have a dictionary of 100 000 tokens, the taxonomy contains 10 000 concepts and the ratio of translation is set to 0.5 – this means that 50 000 tokens will get assigned translation, some of them will share the same definition, some of them will contain only one definition. Thus we also simulate the situation of synonymy where certain words have the same meaning and certain words have no translation at all (despite they should have if we were assigning semantic tags to all patterns).

Before the translation starts, we select a number (200) of pairs from the language model. They are chosen randomly and we do not have control over how many of them also contain semantic codes (but additional parameter can be used to enforce how big portion of these tokens must have semantic codes. If no concept definition is present, it is automatically generated – in our experiment, we set this ratio to be 50%). We thus have the model of our language, the dictionary and the taxonomy used for translation. We proceed to the translation of the previously generated corpus of 20 000 documents. We will read the randomly generated corpus and whenever we encounter a token that belongs to one of the significant pairs, the other pair will be inserted somewhere around the position of the first element. This position is randomly selected, in the range of 1 to 5 tokens – i.e. there can be up to 5 other tokens between the first and the second element of the significant pair.

After the original and the translated corpus were generated, we proceed to the collocation discovery. The reported results are averaged from the batch of 10 runs, where for each run the corpus as well as the language model was completely regenerated.

There exist two modes of collocation search that we discussed in the previous chapter IV.4.4. We can base our search on a better understanding of the collection of documents - for example we have at our disposal corpora of similar or relevant messages that deal with the same topics as the corpus in which we are interested. We can thus build a better approximation (model) of the expected frequencies and their combinations using such data. Or we may simply want to compare a corpus A against a new corpus B and highlight the important differences, which is also possible.

In a perhaps more difficult scenario we do not possess prior information and cannot build the statistical model of the collection and this will be focus of our evaluation. The null hypothesis (to be tested against the real data) will be based on the observed frequencies. It has

the generic form of: $f(AB) = f(A) \times f(B)$ – which says that the expected frequency of a cooccurrence is made of a joint frequency of its components, for more details see previous chapter IV.4.4, p. 92. This assumption is certainly wrong, as language is not random, but at the same time is used successfully in many linguistic tasks where it was proven satisfactory (Hopkins and King 2010; Christopher D. Manning and Schuetze 1999, 237). We will have the opportunity to compare this approach of cooccurrence search against both the randomly generated corpus (for which it should be naturally more suitable) as well as against the corpus based on the real language.

As for the corpus based on real language, we use the publicly available version of the well-known Reuters-21578 "ApteMod" corpus for text categorization. ApteMod is a collection of 10,788 documents from the Reuters financial newswire service, partitioned into a training set with 7769 documents and a test set with 3019 documents. The total size of the corpus is about 43 MB.¹⁵

The distribution of categories in the ApteMod corpus is highly skewed, with 36.7% of the documents in the most common category, and only 0.0185% (2 documents) in each of the five least common categories. In the ApteMod corpus, each document belongs to one or more categories. There are 90 categories in the corpus. The average number of categories per document is 1.235, and the average number of documents per category is about 148, or 1.37% of the corpus.

The procedure for the collocation search is similar with the previous corpus. The selection of 200 significant pairs is done randomly, but we disregard words that occur too often as well as words whose occurrence is rare. The limit is applied to the list of words from the corpus, sorted by frequency and we randomly select significant pairs only from the slice that covers 1.5%-30% of the list. Doing this, we remove words from the first and the second order category of the Zipf law, as well as most of the tokens that occur less than 5 times. The search for the significant cooccurrences is then conducted in the same way as for the first corpus, with average pairs recreated each time and results averaged.

V.2.3.1 Results

We will start by presenting the results from the search on the randomly generated corpus using the observed frequencies null hypothesis. The precision and recall reported here concerns the list of the first 5000 most significant pairs as selected by the algorithm. The chart shows results in bins of 100 items each.

¹⁵ It is also available for download from <http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>, which includes a more extensive history of the data revisions.

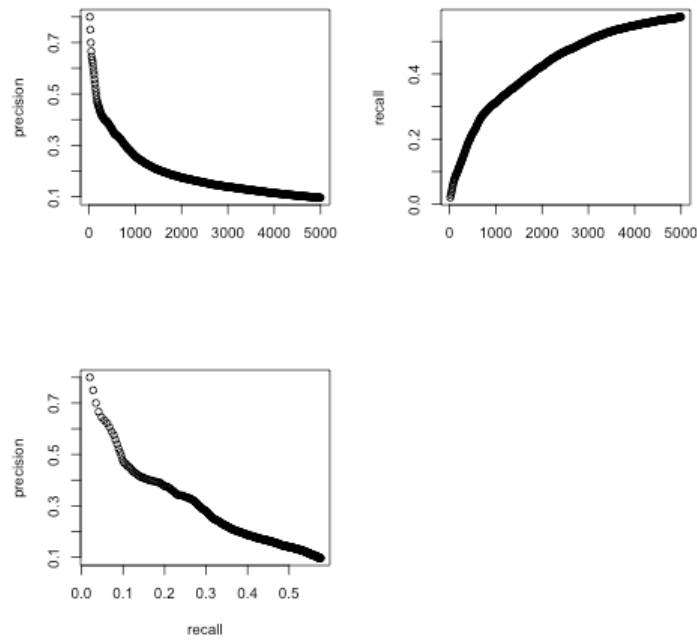


Illustration 29: Results from the processing of the average corpus using the default null hypothesis

The results above include both pairs made of tokens (words) as well as combinations of semes (semantic relations). We can filter out all the semantic relations and look only at

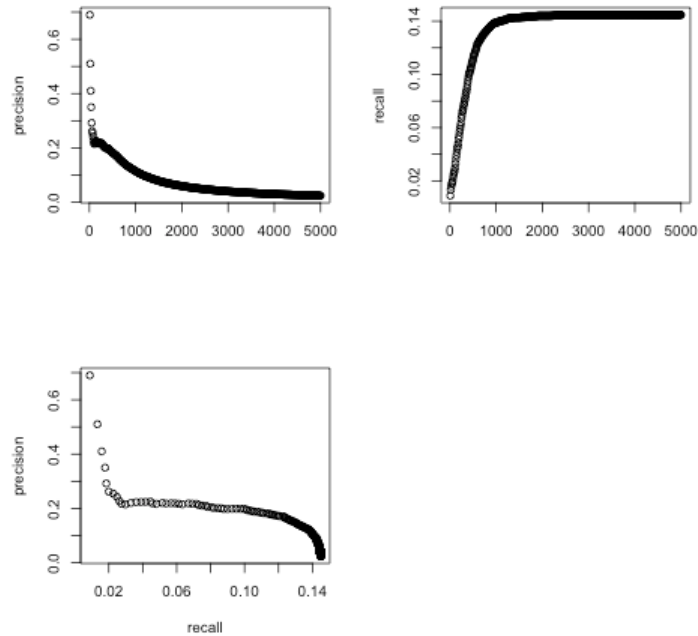


Illustration 30: Random corpus looking only at the words, ignoring the semantic relations

word pairs (ie. to see whether there is a difference between the relationships discovered from the text and those that are based on the enriched semantic information). See

Illustration 30 above.

The previous chart just shows that the portion of word pairs is a fraction of the significant pairs, however the token pairs are clustered towards the beginning of the list. As we continue to retrieve more results, the frequency of word pairs decreases steadily and later we retrieve mostly the semantic pairs. This is correct, as the semantic pairs are more numerous, while for the word pairs the precision decreases slowly. In the overall picture (for all pairs) it decreases sharply and after the first 1000 results it is at 20%, which is rather low.

The second batch of tests was executed on the Reuters21578 corpus. We can observe a slightly different situation. The precision starts at 85% and continues to raise as we harvest

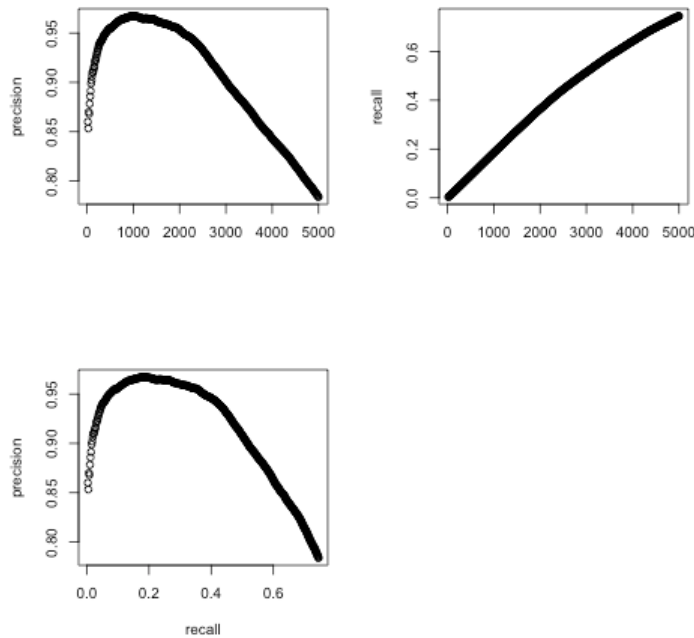


Illustration 31: Precision and recall for the Reuters corpus, in the search for significant pairs based on the expected values computed from the source corpus

more results, to be highest around 1000 pairs. The curve of the recall is almost diagonal which means we are retrieving very high number of positive hits with every step – even if the overall recall is 'only' 70%; thus we are not able to get all the 4883 valid and significant pairs in the list of the 5000 retrieved pairs. Nevertheless, values for both metrics are relatively very high.

If we look at the same run again, but this time filtering out all the semantic pairs and leaving only words as they appear in the corpus, we would see the following results. The recall is very low, about 6% in the first 5000 combinations, which means all the other relevant combinations retrieved in the previous steps are semantic pairs. The distribution of the words pairs is also different from the distribution of the semantic pairs.

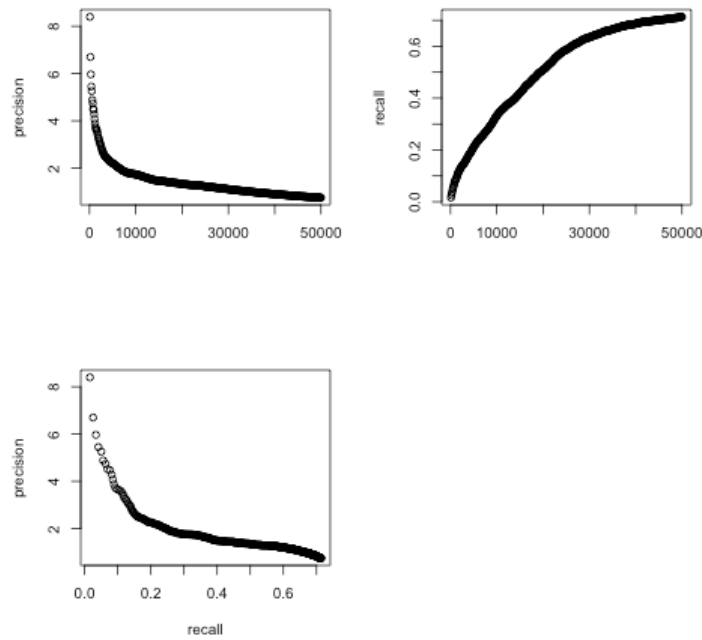


Illustration 32: Precision and recall statistic if we look only at words (ie. ignoring the semantic concept relations altogether). The recall is considerably lower because words make only a small portion of all significant pairs

V.2.3.2 Discussion

We can observe that on the Reuters corpora the first several hundred pairs contain more than 85% of the significant pairs and this precision actually increases long after the first 1000 combinations were discovered. This result is probably due to the different distribution of words that is less random than the first corpus. This is not a bad news for us, but for certain applications also the 70% precision of the automatic null hypothesis search could be acceptable and useful.

If the Reuters corpus exhibits less entropy than the randomly generated one and outliers are easier to detect, the number of pairs that we need to test for statistical significance is also lower. In the case of the randomly generated corpus, where we searched for the cooccurrences in the distance $w=5$, the system had to inspect on average 5 million potential pairs per run. In the case of the Reuters corpus, which contained roughly a similar ratio of translated tokens, the number of pairs that needed to be inspect was lower - slightly more than 1 million pairs per run. In the case of the randomly generated corpus, which is about 40% bigger, the difference is a factor of two and will grow in a linear fashion with the size of the corpus. But in the case of the real language data, it is unlikely that the number of cooccurrences grows linearly because the language is not random and exhibits rather strong consistency.

This last part of the search is arguably the slowest of the whole pipeline because SEMAN takes time in building the indices (and there are two of them if we use the reference collection) so it has to inspect millions of pairs to select the significant ones. To inspect one million of them takes 16 minutes on the reference machine¹⁶, which may not be optimal for large collections and by large we mean corpora of several million documents. However, we could limit the search only to the semes and their combinations. Thus we can process even large corpora because semes usually account for less than 40% of the text. For a small and medium sized collections the processing times can be counted in dozens of minutes or hours. We can also limit the number of inspected pairs by a simple heuristic of frequency.

The time spent in the computation is basically dependent on the parameter f which filters out combinations based on frequency of the individual components. If we want to cover all possible combinations we must set $f=(0.0,1.0)$ but this means we will search too many combinations and most of them too infrequent to be scored as significant. The following chart shows how precision and recall change. The sweet spot in the Reuters21578 corpus was in the range of collocations where both elements are present in at least 0.3% of corpus,

$f(x;0.025)$	F-measure	Ratio of pairs	# of pairs
0	0.614684	1	4506645
0.0001	0.614684	1	4506645
0.0002	0.614684	0.4180382524	1883950
0.0003	0.614684	0.2188592623	986321
0.0004	0.611973	0.1250011927	563336
0.0005	0.611973	0.1250011927	563336
0.0006	0.605791	0.0753893417	339753
0.0007	0.59363	0.0472322537	212859
0.0008	0.576321	0.0306534018	138144
0.0009	0.576321	0.0306534018	138144
0.001	0.544783	0.0205332348	92536
0.0011	0.499859	0.0140958518	63525
0.0012	0.448236	0.0098669853	44467
0.0013	0.41519	0.0070067645	31577
0.0014	0.41519	0.0070067645	31577
0.0015	0.390745	0.0050731753	22863
0.0016	0.350589	0.0037102989	16721
0.0017	0.308377	0.002731522	12310
0.0018	0.308377	0.002731522	12310
0.0019	0.276041	0.0020332198	9163
0.002	0.243025	0.001456294	6563
0.0021	0.203579	0.0009987474	4501
0.0022	0.203579	0.0009987474	4501
0.0023	0.131999	0.0006252989	2818
0.0024	0.071752	0.0002851345	1285

Illustration 33: Number of pairs retrieved for inspection based on parameter f (this run does not represent the averaged f -measure values, but is representative for the ratio of retrieved pairs and the general trend of the f -measure curve)

which translates to roughly 3 documents. As the table below shows, the $f=(0.0003;1)$ will limit the number of pairs that have to be inspected to less than one quarter of all possible pairs. Without impairing recall, we can limit the computation time significantly.

¹⁶ See the machine specification in V.1 The pattern matching mechanism, p. 115

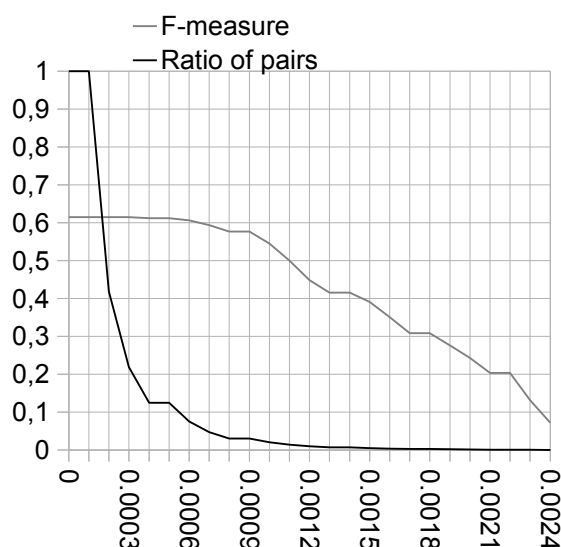


Illustration 34: Plotting the ratio of inspected pairs against the f -measure

But results that were reported for each corpus correspond to $f=(0.0005,0.5)$, in other words we inspect only pairs that are present in at least 0.05% of documents which in the case of the random corpus translates to 10 documents and in the case of Reuters21578 corpora to slightly more than 5 documents. This could partially explain the differences between the two corpora. In the case of the random corpus where the precision starts at 85% and then sharply decreases, it is because some of the ambiguous cases were previously filtered out by the frequency criteria. In the case of the Reuters21578 corpus, we cover more pairs, and less of them were filtered out by the f parameter. So while using the frequency filter can improve the precision and decrease the processing time considerably, we will necessarily lose certain combinations and the recall will inevitably be lower.

Finally the two plots showing the precision and recall of only the words (Illustrations 30 and 32) demonstrate that the system is able to find word pairs but also the semantic relationships. In fact, words represent only 6% of all possible and valid cooccurrences. The semantic pairs were found as well and the performance of that search did not diminish – though the distribution and expected values of the semantic codes is very much different from the distribution of tokens. And this even if certain semantic codes are used in many thousands of other concept combinations. The translation is not always perfect and we face also considerable levels of synonymy – which means that the distribution of the semantic codes is quite a lot different from those of the words. Nevertheless, identification of a great part of the relevant and statistically significant semantic cooccurrence pairs is still feasible.

VI. CONCLUSIONS

The work in this thesis focused on the evaluation of the idea of Universal Semantic Language (USL) in the context of content analysis. We have demonstrated that it is possible to transform the concept of USL into a working application and showed that USL can be used to analyze corpora of textual documents. The real application of this research lays in the ability of the system to translate texts written in the natural language into the collection of semantic codes in which we can search for interesting **semantic relationships**. Such analysis has practical use in many problems that include processing of textual data and extraction of new facts from big corpora.

Firstly, we have reviewed the historical background of the USL and compared it with semantic fields, semantic primitives, and semantic primes and universals of Kant and Wierzbicka. The concept of the lattice which is often present in a description of meaning is not a new one. It is used in many applications of semantic analysis and is well understood. However, the meaning itself is more difficult to formalise. What is possible to do, though, are operational definitions. They consist of a definition of the meaning that is perhaps not universally acceptable, but is applicable to a specific domain. This approach is adopted by researchers in the language processing as well as in content analysis. And the content analysis studies are the selected field of interest for us. Though we did not exclude the possibility that researchers may one day embrace the idea of global and universally acceptable semantic primitives.

Secondly, we have reviewed the basic methodological requirements for the construction of content analysis studies, with a special focus on the operationalisation of concepts (knowledge representation). The second chapter described the elements of content analysis and relation of the theory to the tool we build, but we also saw the practical implications in the example of three content analysis tools that are widely used by the content analysis community. These tools served the purpose of comparison with the application we developed.

We have described the architecture and the components of the new application. It can operate in similar ways to the classical content analysis tools, but also in a special mode which takes advantage of the USL processing. This mode, together with the state-of-the-art natural language processing components, constitute the main difference between SEMAN and the other classical content analysis tools. But building the system itself was not the main aspiration of the thesis. We had to evaluate whether it was possible to extract useful information from the textual corpora with USL.

To achieve this goal, we have evaluated several components. First the pattern matching mechanism which is different than the usual mechanisms employed in similar software tools. We have compared SEMAN against a mature keyword extractor system on a corpus of High Energy Physics documents. In the default configuration our system achieved 0.7 correlation coefficient but after the first 50 and 100 most frequent patterns were manually checked and fixed, the correlation coefficient rose up to 0.87. SEMAN provided information about matches and based on it we could correct many of the obvious mistakes. The correction of the first 100 wrong entries took less than 2 hours and the number of incorrect matches diminished gradually as we processed the list starting with the most frequent patterns. It would be possible to achieve even higher correlation, but only with greater investment of time and energy. The reference system used specialized patterns designed specifically for the target domain. SEMAN, on the other hand, used the generated dictionary with the automatically stemmed entries.

The experiment showed it was possible to use the existing external sources of knowledge representation, such as dictionaries, thesauri, classification systems. The structure of the USL makes it rather easy to convert the existing sources into the language of USL – as was also done in the case of WordNet. Yet WordNet transformation also pointed to limitations of the current form of USL representation. Its strictly linear form (flat data structure of the SEMAN dictionary) is not the ideal mechanism for storage of acyclic knowledge representation data. The editor that we developed for dictionary maintenance makes operations somewhat easier, but the principal problem remained. While it is not impossible to represent the acyclic relations in the form of linear entries, it is not exactly the best solution for WordNet (or other more advanced knowledge data represented in USL).

The pattern matching mechanism is an important component of SEMAN but it was not the only part we tested. Next we have looked at the more serious problem of semantic ambiguity in the process of translation. We have described the experiment of automated document classification in which we used the different sets of data. One more uniform corpus containing longer papers from the domain of High Energy Physics. On this corpus we used the domain specific thesaurus. The other corpus contained twenty thousands news postings and for its translation we have used the general-domain WordNet.

The experiment confirmed that a wrong translation has strongly detrimental effect on the final results of classification. When a certain pattern had several definitions and we used all of them in the classification, we have observed the worst possible performance of all the combinations. This confirms the importance of correct disambiguation. We cannot say yet whether the trend would change, were we to use much bigger document corpora, but it seems reasonable to expect that the entropy introduced by an incorrect translation is very dangerous.

We have therefore shown that the results of classification improved with better word sense disambiguation. Unfortunately, most of the automated word sense disambiguation routines will be too costly in terms of CPU time. Thus we concentrated on a search for a simpler disambiguation techniques that use linguistic information. We have seen that disambiguation based on the POS information did improve results of classification considerably. The scores of document classification that was based purely on semantic features were lower than the plain text feature classification, but the f-measure improved in general from the range of 70% to 90%. In fact, we could see that if we used the semantic

features together with words (when the semantic translation was not available) the document classification produced the best results. This led us to the conclusion that the translation did not introduce more entropy into the source data.

We can assert that for certain applications such an operational mode is more than acceptable. Especially if we process bigger collections of data where we can disregard the individual errors. The second experiment has shown us two important things. First, that we can avoid the costly word sense disambiguation. Even without constructing the complicated dictionary patterns it is possible to achieve levels of accuracy of more than 80%, which is the level of accuracy that was reported satisfactory for at least one of the content analysis tools that we described (TABARI). Secondly, the amount of information translated into the semantic codes, even if we used one of the biggest available semantic networks, accounted for less than 40% of the texts. For these reasons SEMAN allows researchers store plain text and enhance it with the semantic codes (but it can also store only the semantic codes: this operational mode might be more suitable for big collections).

The final part of the evaluation focused on our ability to recognize and retrieve the significant combinations of concepts from the text. For this we constructed two synthetic benchmarks, one modelled randomly with the Zipf distribution which is characteristic for natural languages, the other benchmark used real texts. We have seen that in the case of the random corpus, SEMAN was able to retrieve 70% of significant combinations in the list of the first few hundred results and the precision slowly decreased. In the case of the real text corpus, the performance was much better (85%) and even increased for the first 1000 entries up to the level of 95% to again slowly decrease as we harvested more pairs.

This part of the experiment showed us that it was possible to recognize and retrieve the combinations of semantic codes that are statistically significant – even if we store texts and the semantic codes together and conflate two distributions. But since the words are related to their meaning and if the process of disambiguation did not introduce too much ambiguity into the source data, we could expect the mechanism to work properly. We have concluded from the exercise that SEMAN can recognize and extract the combinations of significant semantic codes from corpora of textual messages. But of course, the task of interpretation of these results is in the hands of humans.

Though we can conclude SEMAN has potential, we shall mention its weak parts. Perhaps the biggest limitation of this type of applications is the cost associated with the production of consistent knowledge representation. The cost might be mitigated if we reuse the existing sources of knowledge, or automatically extract the knowledge from the existing unstructured and semi-structured information resources such as Wikipedia. But even if we do so, it should be noted that many modern information extraction systems need not work with any prescribed knowledge representation. They can extract useful information directly from the corpus of documents, such as Latent Semantic Analysis (LSA). In this respect the approach that we worked on and we described belongs to the paradigm that is not particularly popular in the research. Perhaps one day the pendulum swings back towards the knowledge representation systems with manually prepared data, but in the current stage the very existence and necessity of the dictionary represents the biggest stumbling block.

Provided we have means to overcome this knowledge bottleneck, future research can be focused on the visualisation of the data extracted from the texts. The automated processing of textual data opens doors to many interesting applications and visualisation of even simple relationships may prove very insightful. In the current stage, we focused only on the

discovery of combinations of two concepts. However it might be even more interesting to look at concept trigrams or even much larger groups. And finally, we should definitively try to analyse whether there exist differences between users of different languages in the way which knowledge categories they apply. We should test in a rigorous manner the hypothesis of Anna Wierzbicka which says that there exists something that we call semantic universals, and that people are similar in the way they think. Any confirmation or refutation of such a hypothesis may have far reaching consequences.

VII. BIBLIOGRAPHY

“Assessment and Development of New Methods for the Analysis of Media Content.”
<http://www.restore.ac.uk/lboro/index.php> (Accessed November 21, 2010).

Azar, Edward E. 1980. “The Conflict and Peace Data Bank (COPDAB) Project.” *Journal of Conflict Resolution* 24(1): 143 -152.

Bartsch, Sabine. 2004. *Structural and Functional Properties of Collocations in English*.
Tübingen.

Berelson, Bernard. 1972a. *2 Content analysis in communication research*. New York:
Hafner.

———. 1972b. *2 Content analysis in communication research*. New York: Hafner.

Berelson, Bernard, and Paul F. Lazarsfeld. 1948. *The Analysis of communication content*.
Chicago: University of Chicago Press.

Bickle, John, Peter Mandik, and Anthony Landreth. “The Philosophy of Neuroscience.”
<http://plato.stanford.edu/entries/neuroscience/> (Accessed July 23, 2011).

Cer, Daniel, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning.
2010. “Parsing to Stanford Dependencies: Trade-offs between speed and accuracy.”
In *7th International Conference on Language Resources and Evaluation (LREC 2010)*, http://nlp.stanford.edu/pubs/lrecstanforddeps_final_final.pdf.

Cuilenberg, Jan J., Jan Kleinnijenhuis, and Jan A. de Ridder. 1998. “Artificial Intelligence
and Content Analysis: Problems of and Strategies for Computer Text Analysis.”
Quality and Quantity 22: 65-97.

Cunningham, H. 2005. “Information Extraction, Automatic.” *Encyclopedia of Language
and Linguistics, 2nd Edition*.

Cunningham, H., D. Maynard, K. Bontcheva, and V. Tablan. 2002. “GATE: A framework
and graphical development environment for robust NLP tools and applications.” In
*Proceedings of the 40th Anniversary Meeting of the Association for Computational
Linguistics*,.

Dale, Robert, Hermann Moisl, and Harold Somers. 2000. *Handbook of Natural Language*

- Processing*. 1st ed. CRC Press.
- Evert, Stefan. 2005. "The statistics of word cooccurrences : word pairs and collocations (Ph.D. thesis)." <http://elib.uni-stuttgart.de/opus/volltexte/2005/2371/> (Accessed March 22, 2011).
- Fan, R. E., K. W. Chang, C. J. Hsieh, X. R. Wang, et al. 2008. "LIBLINEAR: A library for large linear classification." *Journal of Machine Learning Research* 9: 1971-1874.
- Firth, J.R. 1957. *Papers in Linguistics 1934-1951*. London: Oxford University Press.
- Forman, George. *A Pitfall and Solution in Multi-Class Feature Selection for Text Classification*.
- . 2003. "An extensive empirical study of feature selection metrics for text classification." *J. Mach. Learn. Res.* 3: 1289-1305.
- George, Alexander L. 1959. *Propaganda Analysis: a Study of Inferences Made from Nazi Propaganda in World War II*. Row, Peterson & Co.
- Giozzo, Alfio, and Carlo Strapparava. 2009. *Semantic Domains in Computational Linguistics*. Springer.
- Goddard, Cliff. 1994. *Semantic and lexical universals : theory and empirical findings*. Amsterdam ; Philadelphia: J. Benjamins.
- Gottschalk, Louis A. 1997. "The unobtrusive Measurement of Psychological States and Traits." In *Text analysis for the social sciences: methods for drawing statistical inferences from texts and transcripts*, Routledge, p. 117-147.
- Gottschalk, Louis A., and R. Bechtel. 1995. "Computerized measurement of the content analysis of natural language for use in biomedical research." *Computer Methods and Programs in Biomedicine* 47: 123-130.
- Harden, Theo. 1983. *An analysis of the semantic field of the German particles "überhaupt" and "eigentlich."* Gunter Narr Verlag.
- Hopkins, Daniel, and Gary King. 2010. "A Method of Automated Nonparametric Content Analysis for Social Science." *American Journal of Political Science* 54(1): 247, 229.
- Hsu, C.-W., C.-C. Chang, and C.-J. Lin. 2003. *A practical guide to support vector classification*. Technical report, Department of Computer Science, National Taiwan University.
- Huberman, B. A., D. M. Romero, and F. Wu. 2009. "Crowdsourcing, attention and productivity." *Journal of Information Science* 35(6): 758-765.
- Joachims, Thorsten. 1998. "Text Categorization with Support Vector Machines: Learning

- with Many Relevant Features.” In *Proceedings of the European Conference on Machine Learning*, Springer.
- Justeson, John S., and Slava M. Katz. 1995. “Technical terminology: some linguistic properties and an algorithm for identification in text.” *Natural Language Engineering* 1(01).
http://www.journals.cambridge.org/abstract_S1351324900000048 (Accessed July 24, 2011).
- Kant, Immanuel. 1992. *Lectures on logic Immanuel Kant translated and edited by J. Michael Young*. CUP.
- Kecskés, István. 2003. *Situation-bound utterances in L1 and L2*. Walter de Gruyter.
- King, Gary. 2003. “10 Million International Dyadic Events.”
<http://hdl.handle.net/1902.1/FYXLAWZRIA>.
- King, Gary, M. Knowles, and S. Melendez. 2010. “ReadMe: Software for Automated Content Analysis.” <http://gking.harvard.edu/readme>.
- King, Gary, and Will Lowe. 2003. “An Automated Information Extraction Tool For International Conflict Data with Performance as Good as Human Coders: A Rare Events Evaluation Design.” *International Organization* 57(3): 617-642.
- Koenig, Thomas. “CAQDAS Comparison.”
http://www.restore.ac.uk/lboro/research/software/caqdas_comparison.php
 (Accessed November 21, 2010).
- Kolhatkar, Varada. 2009. “An Extended Analysis of a Method of All Words Sense Disambiguation.” University of Minnesota.
<http://www.d.umn.edu/~tpederse/Pubs/varada-thesis.pdf>.
- Krippendorff, Klaus. 2004a. 2. *Content analysis: An introduction to its methodology*. Thousand Oaks: Sage.
- . 2004b. *Content analysis: An introduction to its methodology*. Thousand Oaks: Sage.
- Laurance, Edward J. 1990. “Events data and policy analysis.” *Policy Sciences* 23(2): 111-132.
- Leininger, Kurt. 2000. “Interindexer consistency in PsycINFO.” *Journal of Librarianship and Information Science* 32(1): 4 -8.
- Lejeune, Christophe. 2008. “Au fil de l’interprétation. L’apport des registres aux logiciels d’analyse qualitative.” *Revue Suisse de Sociologie* 34(3): 593-603.
- . 2009. “Méthodes qualitatives informatisées.”
<http://analyses.ishs.ulg.ac.be/logiciels/index.html> (Accessed January 24, 2010).

- Lowe, Will. 2003. *Content Analysis Software: A Review*. Identity Project, Weatherhead Center for International Affairs, Harvard University.
<http://www.wcfia.harvard.edu/misc/initiative/identity>.
- . 2008. "Understanding Wordscores." *Political Analysis* 16(4): 371, 356.
- . "Yoshikoder: An Open Source Multilingual Content Analysis Tool for Social Scientists." <http://www.yoshikoder.org/courses/apsa2006/apsa-yk.pdf>.
- MacMillan, Katie. 2005. "More Than Just Coding? Evaluating CAQDAS in a Discourse Analysis of News Texts." *Forum: Qualitative social research* 6(3). <http://nbn-resolving.de/urn:nbn:de:0114-fqs0503257> (Accessed November 21, 2010).
- Manning, Christopher D., and Hinrich Schuetze. 1999. *Foundations of Statistical Natural Language Processing*. MIT.
- McClelland, Charles. 1999. *World Event/Interaction Survey (WEIS) Project, 1966-1978 [Computer file]*. Ann Arbor, MI.
- Medelyan, Olena. 2009. "Human-competitive automatic topic indexing." University of Waikato. <http://hdl.handle.net/10289/3513>.
- Medelyan, Olena, and I. H. Witten. "Measuring inter-indexer consistency using a thesaurus." In *6th ACM/IEEE-CS Joint Conf. on Digital Libraries*, Chapel Hill, NC, USA: ACM Press, p. 274-275.
- Navigli, Roberto. 2009. "Word sense disambiguation: A survey." *ACM Comput. Surv.* 41(2): 1-69.
- Neuendorf, Kimberly A. 2001. *The Content Analysis Guidebook*. 1st ed. Sage Publications, Inc.
- "PCAD 2000." <http://www.gb-software.com/pcad2000.htm> (Accessed November 21, 2010).
- Pedersen, Ted, and Varada Kolhatkar. 2009. "WordNet::SenseRelate::AllWords - A Broad Coverage Word Sense Tagger that Maximizes Semantic Relatedness." In *Proceedings of the North American Chapter of the Association for Computational Linguistics*, Boulder, CO., p. 17-20.
- Phillips, David P. 1979. "Suicide, Motor Vehicle Fatalities, and the Mass Media: Evidence Toward a Theory of Suggestion." *American Journal of Sociology* 84(5): 1150-1174.
- . 1983. "The Impact of Mass Media Violence on U.S. Homicides." *American Sociological Review* 48(4): 560-568.
- Rajman, Martin. 2007. *Speech and language engineering*. 1st ed. Lausanne ;Boca Raton: EPFL Press ;Distributed by CRC Press.

- “SQLAlchemy - The Database Toolkit for Python.” <http://www.sqlalchemy.org/> (Accessed November 17, 2010).
- “SVM-perf: Support Vector Machine for Multivariate Performance Measures.” http://www.cs.cornell.edu/People/tj/svm_light/svm_perf.html (Accessed April 25, 2010).
- Saeed, John. 2009. *Semantics*. 3rd ed. Malden Mass.: Wiley-Blackwell.
- Sampson, Geoffrey. 2003. “The Oxford Handbook of Computational Linguistics.” In , p. 333-336. <http://llc.oxfordjournals.org>.
- Schmolze, James G, Bolt Beranek, and Newman Inc. 1985. “An overview of the KL-ONE knowledge representation system.” *COGNITIVE SCIENCE* 9: 171--216.
- Schrodt, Philip A. 2009. “TABARI. Textual Analysis by Augmented Replacement Instructions Version 0.7.” 9024. <http://web.ku.edu/keds/tabari.dir/tabari.manual.0.7.3b3.pdf> (Accessed July 23, 2011).
- Schrodt, Philip A. 2006. “Twenty Years of the Kansas Event Data System Project.” *The Political Methodist* 14(1): 2-8.
- Schrodt, Philip A., and Deborah J. Gerner. *Analyzing International Event Data: A Handbook of Computer-Based Techniques*. <http://eventdata.psu.edu/papers.dir/AIED.Preface.pdf>.
- Sebastiani, Fabrizio. 2002. “Machine Learning in Automated Text Categorization.” *ACM COMPUTING SURVEYS* 34: 1--47.
- Shapiro, Gilbert, Timothy Tackett, Philip Dawson, and John Markoff. 1998. *Revolutionary demands: a content analysis of the Cahiers de doléances of 1789*. Stanford University Press.
- Smadja, Frank. 1993. “Retrieving collocations from text: Xtract.” *Computational linguistics* 19: 143-177.
- Sowa, John. 2000. *Knowledge representation : logical, philosophical, and computational foundations*. Pacific Grove: Brooks/Cole.
- Stone, Philip. 2001. “Note Introducing Server Version of General Inquirer -- from inquirer blog.” http://www.wjh.harvard.edu/~inquirer/server_blognote.html (Accessed November 21, 2010).
- Szabo, Gabor, and Bernardo A. Huberman. 2010. “Predicting the popularity of online content.” *Communications of the ACM* 53(8): 80.
- Titscher, Stefan, Bryan Jenner, and Michael Meyer. *Methods of text and discourse analysis*.

- Turmo, Jordi, Alicia Ageno, and Neus Català. 2006. "Adaptive information extraction." *ACM Comput. Surv.* 38(2): 4.
- Violi, Patrizia. 2001. *Meaning and experience*. Indiana University Press.
- Weida, Robert. 1991. *Knowledge Representation and Reasoning with Definitional Taxonomies*. Department of Computer Science Columbia University. Technical report. http://www.google.ch/url?sa=t&source=web&cd=4&sqi=2&ved=0CCkQFjAD&url=http%3A%2F%2Fwww.cs.columbia.edu%2F~library%2FTR-repository%2Freports%2Freports-1991%2Fcucs-047-91.ps.gz&rct=j&q=%22terminological%20reasoner%22%20winograd&ei=MhfcTKzhCuKL4gbXycTKCA&usg=AFQjCNHZZTal4soYucsflrRb95mwtanihA&sig2=pwG2e_s1Q72wDEkCg3auHA&cad=rja (Accessed November 11, 2010).
- West, Mark D. 2001. *Theory, method, and practice in computer content analysis*. Greenwood Publishing Group.
- De Wever, B., T. Schellens, M. Valcke, and H. Van Keer. 2006. "Content analysis schemes to analyze transcripts of online asynchronous discussion groups: a review." *Comput. Educ.* 46(1): 6–28.
- White, Marilyn Domas, and Emily E. Marsh. 2006. "Content Analysis: A Flexible Methodology." *Library Trends* 55(1): 22-45.
- Wierzbicka, Anna. 1996. *Semantics : primes and universals*. Oxford [England] ; New York: Oxford University Press.
- Williams, Geoffrey. 2003. "Les collocations et l'école contextualiste britannique." In *Les Collocations: analyse et traitement*, eds. F. Grossmann and A. Tutin. Amsterdam: De Werelt, p. 33-44.
- Wittgenstein, Ludwig. 1998. *Filosofická zkoumání*. 2nd ed. Praha: Filosofia.
- Yarowsky, David. 1992. "Word-sense disambiguation using statistical models of Roget's categories trained on large corpora." *COLING* 14: 454-460.

VIII. APPENDICES

VIII.1 ILLUSTRATION INDEX

Illustration 1: Tree of Porphyry.....	8
Illustration 2: Visualisation of the High-Energy physics taxonomy written in the USL. Some components are not attached to any word (like 0003w) and the FCA discovered two places where the entries miss connections (the empty boxes).....	11
Illustration 3: A-box visually, using the input from the T-box, notation of existential graphs by C.S. Peirce.....	13
Illustration 4: The change of the INTELLECTUAL field's structure in German at around 1200 AD (left) and at around 1300 AD (right). Originally appeared in Strapparava, p. 15.	17
Illustration 5: Example document analysis, showing details of how the document was parsed - with the absolute frequencies of tokens and more details (down the page; not visible).....	25
Illustration 6: An example output from the General Inquirer processing a news from Burma polls. Categories are in the left column, matched words in the right one.....	58
Illustration 7: Correlation between computer and hand coded data, in the case of first column, the same scheme (WEIS) was used. In the case of the second column, the different coding schemes are used and therefore there is an additional level of discrepancy.....	62
Illustration 8: List of types that TABARI recognizes.....	64
Illustration 9: An example coding of the diplomatic communication in the 'Kingdom of the Ring', TABARI allows for very quick edits and recoding of the whole corpus.....	67
Illustration 10: Example of the aggregation of the categorical scoring scheme (WEIS) into the numerical values coding Scheme (Goldstein).....	68
Illustration 11: Comparison of KEDS and TABARI system on a large corpus of Levant Data.....	69
Illustration 12: Dictionary reports for the distribution of categories.....	74
Illustration 13: The interface to Yoshikoder with the visible dictionary tree and the highlighted matches and the concordance report at the bottom.....	75
Illustration 14: Unified Frequency Report for two documents.....	75
Illustration 15: Graphical view of the dictionary inside Yoshikoder and also the xml structure of the data.....	76
Illustration 16: General overview of GATE, the red area indicates the area of components which can be currently used by SEMAN.....	80
Illustration 17: Components of the ANNIE subsystem and the flow of a document.....	81
Illustration 18: Database schema of the document storage; built around tokens and their translation.....	90

Illustration 19: Comparison of p-values for measures from the significance of association group, using Fisher as a reference point (labels on the axes refer to $-\log_{10} p_v$). Source: Evert, 2005, p. 111. G2 is the log-likelihood and “t” is the Students t-test. These graphs show that X2 statistic favorized rare events, while the t-test gave undesired advantage to frequent events. The other two tests results were much closer to the Fisher's p-value.....	104
Illustration 20: The dictionary editing session, with some special functions activated (autocompletion of semantic codes).....	107
Illustration 21: Another set of commands of the editor, on the right side are visible the maintenance routines for more complex operations.....	108
Illustration 22: Example debugging session of the document analysis - on the right side a few processing steps are selected and executed as one. The window at the bottom shows log messages as the analysis proceeds.....	109
Illustration 23: Example document analysis, showing which tokens were found - with their absolute frequencies.....	110
Illustration 24: Risk Ratio is the statistical method implemented in Yoshikoder. It tests whether the difference between two documents are significant.	111
Illustration 25: Percentage of 'yet-unseen matches' discovered as we continue processing a corpus of 20.000 documents.....	124
Illustration 26: Comparison of the classification results that were based on different datasets.....	136
Illustration 27: F-measure for the document classification of 20 Newsgroups.....	137
Illustration 28: Aggregate view on the document classification for 20 Newsgroups corpus... 138	
Illustration 29: Results from the processing of the average corpus using the default null hypothesis.....	145
Illustration 30: Random corpus looking only at the words, ignoring the semantic relations... 145	
Illustration 31: Precision and recall for the Reuters corpus, in the search for significant pairs based on the expected values computed from the source corpus.....	146
Illustration 32: Precision and recall statistic if we look only at words (ie. ignoring the semantic concept relations altogether). The recall is considerably lower because words make only a small portion of all significant pairs.....	147
Illustration 33: Number of pairs retrieved for inspection based on parameter f (this run does not represent the averaged f-measure values, but is representative for the ratio of retrieved pairs and the general trend of the f-measure curve).....	148
Illustration 34: Plotting the ratio of inspected pairs against the f-measure.....	149

VIII.2 BIBLIOGRAPHY ON SEMAN

1. Smetáček, V. (2008). Systém SEMAN. In Elektronické studijní texty. Dostupné z: <http://texty.jinonice.cuni.cz>. [cit. 2009-03-12]

2. Smetáček, V. (1988). Uživatelské chody báze BALEX. Metodický zpravodaj československé soustavy VTEI, 1988, roč. 16, č. 3, s. 3-52.
3. Uličný, O. (1988). Rozvoj metody SEMAN v rámci výzkumných úkolů VTEI. Knižnice a vedecké informácie, 1988, roč. 20, s. 59-61.
4. Smetáček, V. (1987). Tezaurus sémů. Automatizovaná báze lexikálních jednotek BALEX : aktuality a materiály, 1987, č. 7, nestr.
5. Smetáček, V., Mikesková, M. (1987). O báze BALEX a metóde SEMAN. Knižnice a vedecké informácie, 1987, roč. 19, č. 5, s. 230-232.
6. Uličný, O. (1987). Automatizovaná tvorba tezauru s využitím metody SEMAN. Československá informatika, 1987, roč. 29, č. 1, s. 16.
7. Smetáček, V. (1986?). Obsahová analýza literárního textu s pomocí sémantického kódu : (první verze). 1986?, 60 s.+přil. Strojopis.
8. Smetáček, V., Kubešová, M. (1986). Budování a možnosti využití báze lexikálních jednotek BALEX. In Lingvistické metody a automatizované informační systémy. Praha : Dům techniky ČSVTS, 1986, s. 96-103.
9. Smetáček, V., Nyklová, A., Uličný, O. (1986). Automatizovaná tvorba tezaurů. In Lingvistické metody a automatizované informační systémy. Praha : Dům techniky ČSVTS, 1986, s. 103-109.
10. Nyklová, A. (1986). Automatické vytváření slovníku typu tezauru ze souboru lexikálních jednotek (BALEX-ATEZ) : Provozní dokumentace. Automatizovaná báze lexikálních jednotek BALEX : aktuality a materiály, 1986, č. 1, 49 s.+přil.
11. Jonák, Z. (1986). Systém lingvistického zabezpečení metodou SÉMAN. In Lingvistické metody a automatizované informační systémy. Praha : Dům techniky ČSVTS, 1986, s. 118-125.
12. Smetáček, V. (1985b). Prvky umělé inteligence v lingvistickém zabezpečeníází dat. In Informatika 90. let. Praha : Dům techniky ČSVTS, 1985, s. 45-50.
13. Smetáček, V. (1985a). Experimentální ověření vlivu hodnot jednotlivých proměnných na výsledky procedury ATEZ. Automatizovaná báze lexikálních jednotek BALEX : aktuality a materiály, 1985, č. 9, s. 2-16.
14. Uličný, O., Webr, J. (1985). K problematice automatizované tvorby a aktualizace tezauru. Automatizovaná báze lexikálních jednotek BALEX : aktuality a materiály, 1985, č. 6, s. 20-37.
15. Čermáková, A., Smetáček, V., Uličný, O. (1985). Automatické vytváření slovníku typu tezauru ze souboru lexikálních jednotek nebo znaků klasifikací (BALEX-ATEZ) : Návrh technologie. Automatizovaná báze lexikálních jednotek BALEX : aktuality a materiály, 1985, č. 7, 18 s.+přil.
16. Smetáček, V. (1984b). Automatizovaná tvorba tezauru s pomocí metody sémantického analyzátoru. In Selekčné jazyky '84 : (zborník zo seminára konaného v dňoch 20.-21. júna 1984 v Bratislave). Bratislava : Slovenská technická knižnica, 1984, s. 60-65.
17. Smetáček, V. (1984a). Sémantický analyzátor : (experimentální ověřování). Olomouc : Univerzita Palackého, 1984. 296 s.

18. Uličný, O., Webr, J. (1984). K problematice automatizované tvorby a aktualizace tezauru. *Československá informatika*, 1984, roč. 28, č. 6, s. 161-167.
19. Smetáček, V., Webr, J. (1983). Možnost automatického zjišťování stupně obsahové příbuznosti lexikálních jednotek přirozeného selekčního jazyka. *Československá informatika*, 1983, roč. 25, č. 7/8, s. 197-204.
20. Smetáček, V. (1982b). *Sémantický analyzátor : základní pojmy a prvky (úvod do problematiky)*. Olomouc : Univerzita Palackého, 1982. 189 s.
21. Uličný, O. (1982). Struktura sémantického analyzátoru jako prostředku sémantické analýzy textu. In *Využití lingvistických přístupů v informatice*. Městský seminář, Praha, 10.-11. června 1982. Praha : Dům techniky ČSVTS, 1982, s. 81-87.
22. Smetáček, V. (1982a). SEMAN - experimentální automatizovaný nástroj obsahové analýzy textů v přirozeném jazyce. In *Využití lingvistických přístupů v informatice*. Městský seminář, Praha, 10.-11. června 1982. Praha : Dům techniky ČSVTS, 1982, s. 55-63.
23. Jonák, Z. (1982). Experimentální ověření sémantického analyzátoru při automatickém indexování. In *Využití lingvistických přístupů v informatice*. Městský seminář, Praha, 10.-11. června 1982. Praha : Dům techniky ČSVTS, 1982, s. 64-72.

VIII.3 INSTALLATION INSTRUCTIONS

SEMAN is released as open-source software under the BSD license. The installation instructions and documentation for SEMAN is available at the following address:

<http://code.google.com/p/newseman/>

The source code repository contains a branch marked “thesis” with the code and data that were used for writing the thesis.

