Comenius University, Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics

Natural Language Processing with Application to Slovak Language

Rigorous thesis

Mgr. Michal Malý

Bratislava, 2011

Abstrakt

Spracovanie prirodzeného jazyka má v dnešnom informačnom svete mnoho využití. Táto práca ukazuje niektoré zaujímavé aplikácie spracovania prirodzeného jazyka a to najmä s použitím na slovenský jazyk.

Najprv sú vysvetlené základné lingvistické koncepty a koncepty teórie formálnych jazykov. Práca takisto obsahuje základné informácie o slovenskom jazyku (o jeho morfológii a syntaxi).

Potom sú predstavené metódy (všetky založené na spracovaní textového jazyka) spolu s ich aplikáciami a príkladmi. Sú tiež ukázané ich limitácie. Tiež je popísaná originálna metóda pre stemovanie (zoskupovanie podľa koreňov slov), a jej porovnanie s niektorými známimi stemmermi .

kľúčové slová: spracovanie prirodzeného jazyka, slovenský jazyk, gramatická indukcia, strojový preklad

Abstract

Nowadays, natural language processing has a lot of uses in our information world. This work shows some interesting applications of natural language processing, mainly with application to Slovak language.

First, basic linguistic concepts and concepts of theory of formal languages are explained. The work also contains basic informations about Slovak language (its morphology and syntax).

Then methods (all of them based on textual language processing) are introduced, together with applications and examples. Also their limitations are presented. Also an original method for stemming (grouping words according to their root) is described.

keywords: natural language processing, Slovak language, grammatical induction, machine translation

Contents

Introduction and Overview

This work deals with textual methods of natural language processing. In the first chapter, basic language concepts are presented – what is the language, what is the definition of natural language, how can be languages related to each other, and how we learn the language.

Second chapter concerns formal languages in order to prepare theoretical grounds. The third chapter introduces the reader to the computational linguistics and natural language processing and the fourth chapter explains morphology and syntax of Slovak language. The fifth chapter introduces machine translation and presents an example of this method. The sixth chapter presents various applications like chatterbots or mechanical rephrasing. The last chapter presents my own method – using a grammar induction to create a stemming dictionary for Slovak language.

Chapter 1

Language

Language is one of the most interesting phenomena in human abilities. It serves as a foundation for ideas formulation and their communication – we use it to speak to others and also to speak to ourselves in our minds. It is known that animals (e.g. bees) also use some – much simpler – form of language. Human language is a broad and complex system. It probably emerged as a support for cooperation, and was supported by growth of the brain. Since then it has noticeably evolved and is deeply engraved in our culture. The difference between animal language and human language is vast, therefore it is speculated to be caused by brain differences between humans and animals. Our ability to use a complex language is due to Broca's area, Wernicke's area, Primary Auditory Cortex, Supramarginal gyrus, Angular gyrus, and possibly other areas. Some use of language is linked to theory of minds – our ability to understand other's actions, which seems to be rooted in mirror neurons.

A language always uses some kind of symbols to convey meaning. It also needs a set of rules which describe how these symbols can be used. This manipulation with symbols is cognitively demanding.

The observation of symbolic manipulations in the language also gave an impulse for the idea of formal languages. Theory of formal languages concerns primarily word construction, operations with sets of words, and examines mathematical properties of these constructions. To be distinguished from a formal language, the term natural language is sometimes used for a human language.

An artificial (constructed) language is a language created deliberately for some specific purpose. It could be intended for simplifying communication, for example Esperanto was created to be easy to learn and politically neutral. A language could be also created as an experiment in linguistics. Lojban was created to reflect the principles of logic, and to test the Sapir–Whorf hypothesis. This hypothesis states that the language, because of its limits, influences our thought and sometimes also our behavior. A language can be created as an art, e.g. as a fictional language in a fantasy book, movie, or a computer game – Tolkien's Quenya and Sindarin, former inspired by Finnish and latter by Welsh; Klingon language in Star Trek [1], Simlish in The Sims game . A computer language is a language created to communicate with a computer, e.g. to express an algorithm (programming language), to specify the content or layout of a web page (e.g. HTML, CSS), or more broadly, to annotate text with some kind of meta-information (markup language, e.g. XML). Lastly, a language is also used for a specific form of natural language, e.g. English language.

1.1 Natural language

Natural language is any language which evolved in a natural way to communicate between humans. It was first spoken and gained a written form only after a long time. Only explanation rather than construction of its rules is attempted, with possible exception of small regulations by an official body (e.g. Russian Language Institute of the Russian Academy of Sciences, Rat für deutsche Rechtschreibung, Ľudovít Štúr Institute of Linguistics of the Slovak Academy of Sciences).

Natural languages are subject of study of linguistics. Exact definition of the term (natural) language varies among different linguists. We do not know and understand all aspects of natural languages. By comparing different natural languages and by executing linguistic experiments we try to learn about them. This is somewhat a paradox: a normal human child is able to learn a language; however we do not understand this process well and we are not able to implement satisfactory understanding of a natural language into computers. Every one of us is an everyday user of some natural language. Despite of this, one could be perplexed by a "simple" linguistic question, for example, what is the meaning of the word "of"? We use it almost intuitively.

[1]which is probably the most developed fictional language with many "alien", uncommon features. Even an opera with the name " 'u' " ("universe") was written in Klingonaase.

Every pre-school child is able to use grammar, even if it cannot not explicitly state the rules.

1.2 Language family

The languages of the world can be divided into language families. A language family is a group of phylogenetically related languages – derived from a common ancestor language (called proto-language, original language, common language, or basic language). In some language families this original language can be reconstructed to some extent by a systematic comparison of individual languages. The phylogenetic relationships between languages is studied by historical and comparative linguistics.

Two or more different languages can influence each other also because of the geographical proximity and an intensive language contact between the speakers of the languages. This group of languages is called *linguistic area* or *"Sprachbund"* (from German). The languages may be phylogenetically unrelated (e.g. Slovak and Hungarian) and thus the secondary similarities must be distinguished from the primary – phylogenetic – relationship. In many cases it is possible to know what languages belong to the given language family, however it might be more difficult to establish the relationship (the phylogenetic tree) of the family.

The phylogenetic classification can be guessed from the structural features (such as inflection, agglutination, ergativity, vocal harmony, tonal language, etc.), however this may or may not indicate a common proto-language. Geographical and historical information can be used to detect linguistic areas and decide whether the similarities of the languages evolved through the contact or are the result of the same phylogenetic origin.

Sometimes the dialects vary from one location to a neighboring one only slightly so the speakers understand each other without problems, however, the greater the distance becomes, the greater are the accumulated differences – at a certain distance they are not intelligible. It is a question where to draw the line – the term *dialect continuum* is used in this situation. In many cases, the language definition depends also on political or social conditions.

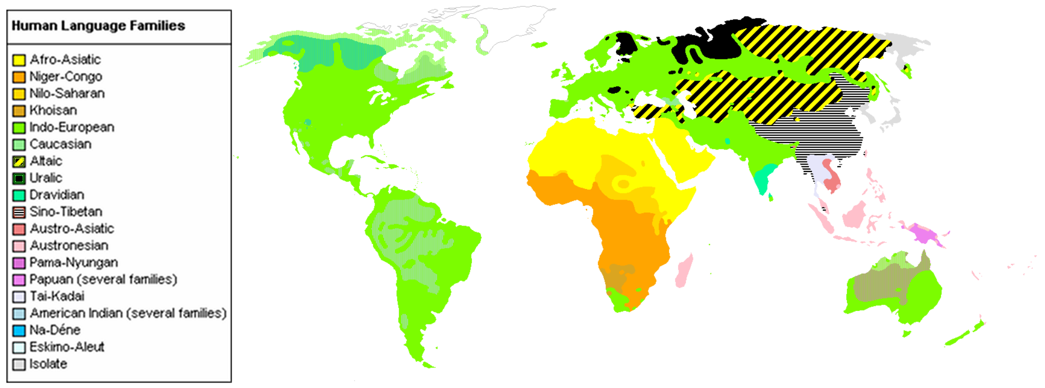

Figure 1.1: Distribution of the major language families, from [6].

1.3 Language learning/acquisition

Language learning hypothesis is closely related to language origins. Some see language as innate to a large extend, some think that we learn most of the language through social interaction. The main source of language acquisition knowledge is the observation of children.

Language development among children is gradual. First comes babbling, responding to its name, responding to human voice and its tone. The child then repeats isolated words, mainly nouns or phrases. Then it proceeds to inflection and simple grammatical constructions and can follow simple commands. Later it can use simple sentences with a verb and use pronouns. It can reason about simple things. Older children can grasp more difficult concepts and use complex sentences [23].

Language development is almost complete at about 8 years, however, we can learn new words or improve our language use during the whole lifetime. For example, Japanese children master complicated Japanese honorific speech and politeness at the teen age.

1.3.1 Theories of language acquisition

One of the most prominent theories is the *universal grammar* theory associated with Noam Chomsky, who began to introduce formal concepts into linguistics[3]. In a nutshell, the theory states that some rules – grammar restrictions – are innate. When a child is learning a language, a corresponding part of the brain ("Language Acquisition Device") recognizes "parameters" of the concrete language (such as the usual word order – Subject-Verb-Object, Subject-Object-Verb, etc.). The restrictions help the child to choose from many hypothetically possible grammars and the innate cues help to recognize the principles of the language. The current version of Chomsky's idea is called "minimalist program" proposed in [4].

1.3.2 Language acquisition modelling

Some argue, that the symbolic approach of generative grammars cannot fully account mistakes and errors which a child makes when learning a language.

Connectionist models usually focus on the word prediction task (started by the work [10], which can be modelled also by unsupervised approach [12]. However also non-adjacent dependencies are modelled [28, 13].

There exist also computational models of language acquisition, e.g. ADIOS (Automatic Distillation of Structures). For Slovak language, its modification AMIGOS (Adios' morphologically-intensive-grammar-oriented specialization) was developed in [39].

Chapter 2

Formal languages

A formal language is a set of finite words (i.e. of finite length) over some alphabet. Instead of the term “word” sometimes the term “string” is used. Depending on the context and application, the alphabet can consist of letters, symbols, or tokens. Formal languages are suitable for a mathematically precise description of the words and the language, for example a formal programming language specification can be useful to avoid ambiguity and confusion. Although the theory of formal languages uses terms borrowed from natural languages – language , word , alphabet , grammar , etc. – these are precisely defined mathematical constructs and their meaning can differ slightly from that of natural languages.

2.1 Definitions

Definition 1. An alphabet is a finite nonempty set of symbols.

Definition 2. A word is a finite sequence of symbols from the alphabet. An empty word is denoted by the symbol ε.

Definition 3. A (formal[1]) language is a set of words.

The formal languages theory studies mostly syntactical – structural properties of a formal languages. A formal language can be specified using

- mathematical notation – set specification (e.g. $L = \{a^n \mid n \text{ is a prime}\}$

[1]I will sometimes drop the adjective “formal” if the context is clear.

- set of rules describing, how it is possible to generate words – a (formal) grammar
- an automaton (a theoretical machine), which for a given word decides, whether this word belongs to the language – it is “accepted”
- a regular expression, which matches words of this language

An important part of formal languages theory concerns of relations between these specification formalisms. I will introduce some of these formalisms and some problems in this chapter.

2.2 Operations on words and languages

For convenience, I will introduce some operations on words and (almost) analogical operations on languages.

Definition 4. Let x and y be words. Then $x.y$ (or sometimes only xy) means word concatenation, i.e. if x is a sequence of symbols $a_0, a_1, \dots, a_m$ and $y = b_0, b_1, \dots, b_n$, then $x.y = a_0, a_1, \dots, a_m, b_0, b_1, \dots, b_n$.

Definition 5. The powers are defined as $x^0 = \varepsilon, x^{(i+1)} = x.x^i$.

Definition 6. The iteration is defined by

$$x^* = \cup_{i=0}^{\infty} x^i$$

For languages, analogous operations can be defined:

Definition 7. Let L_1 and L_2 be two languages. Then $L_1.L_2$ (or sometimes only L_2L_2) is the language consisting of all words of the form vw where v is a word from L_1 and w is a word from L_2.

Definition 8. The potency is defined by $L^0 = \{\varepsilon\}, L^{(i+1)} = L.L^i$.

Definition 9 (Kleene iteration and the “plus” operator)**.** The Kleene iteration is defined by

$$L^* = \cup_{i=0}^{\infty} L^i$$

In addition, let us define

$$L^+ = \cup_{i=1}^{\infty} L^i$$

In addition to these operations we can use the standard set operations $\cup, \cap, \subset$ and other. The negation $\neg L$ is defined with respect to all possible words of the given alphabet. These operators could be used also for alphabets.

2.3 Formal grammar

Definition 10. Grammar is quadruple (N, T, P, σ), where N is a set of nonterminals, T is a set of terminals, P is a set of rules, σ is a starting nonterminal. The rules of P have to be of the form $(N \cup T)^*N(N \cup T)^* \rightarrow (N \cup T)^*$.

Definition 11. The relation $\Rightarrow_G$ of one-step derivation in G is defined by

$$x \Rightarrow_G y \quad \Longleftrightarrow_{\text{def}} \quad \exists u, v, p, q \in (T \cup N)^* : x = u.p.v \wedge p \rightarrow q \in P \wedge y = u.q.v$$

The relation $\Rightarrow_G^*$ is the reflexive transitive closure of the relation $\Rightarrow_G$.

Definition 12. (Language generated by the grammar) The language generated by the grammar G is denoted $L(G) = \{w \mid \sigma \Rightarrow_G^* w\}$.

2.4 Grammars of different power

The form of the rule can be further restricted. According to the level of restriction, a hierarchy of these grammar classes can be build. “Hierarchy” means that each class contains the preceding classes.

regular grammar allows only rules of the form $N \rightarrow T^*N$ or $N \rightarrow T^*$, i.e. the nonterminal can be at most one and must be at the end of the rule “body” and the “head” is restricted to one nonterminal.

context-free grammar allows only rules of the form $N \rightarrow (N \cup T)^*$, i.e. the head of the rule must be only one nonterminal, without the “context” (what gave the name of this class).

context-sensitive grammar allows only rules of the form $x\alpha y \rightarrow x\beta y$, where $x, y \in (N \cup T)^*, \alpha \in N, \beta \in (N \cup T)+$, i.e. the “head” of the rule can specify context, which must be preserved. Moreover, the nonterminal β must not be erased. As a special exception, the rule $\sigma \rightarrow \varepsilon$ is permitted[2].

unrestricted grammar allows rules of any form, i.e. $(N \cup T)^*N(N \cup T)^* \rightarrow (N \cup T)^*$.

[2]This enables the grammar to generate an empty word . This technicality makes context-sensitive languages a superclass of context-free languages; without this addition only a context-free languages not containing ε would be contained.

2.5 Chomsky hierarchy

An interesting finding [29] is that the hierarchy of grammar classes, which seems arbitrary at first look, corresponds to the hierarchy of automaton classes. The regular languages class (languages generated by a regular grammar) contains all finite languages (finite sets of words). A regular language can be recognized by a Finite State Automaton. The context-free languages class corresponds to the class of languages accepted by Non-deterministic Push-down Automaton. A context-sensitive language can be generated by a context-sensitive grammar or accepted by a Linear Bounded Automaton. The class of all languages generated by unrestricted grammar corresponds to the languages recognized by a Turing machine; or in another words, to the class of recursively enumerable languages. For the respective definitions of the machines (FSA, NPDA, LBA, Turing machine) I refer the reader to [15].

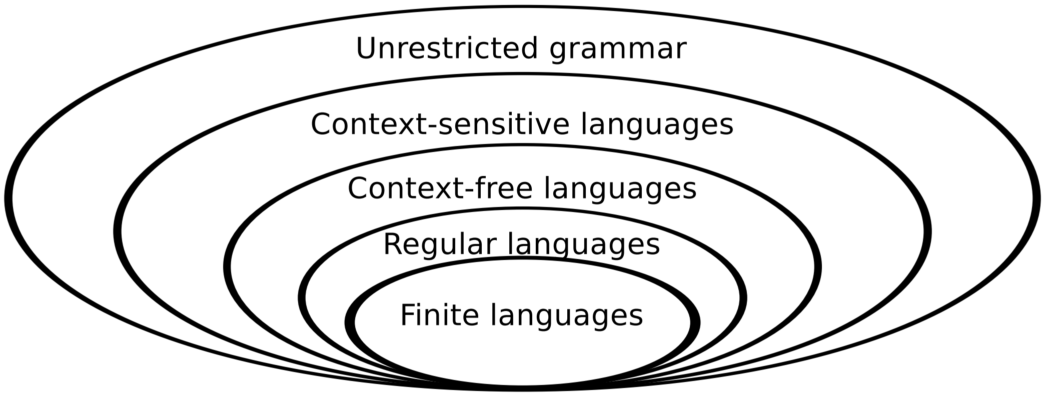

Figure 2.1: Chomsky hierarchy of languages

It is necessary to understand this diagram as a depiction of a series of theorems: every set is proper; it is possible to prove that the lower class language can be expressed by the means of the upper class, and that there exist languages in the upper class that cannot be expressed by the means of the lower class.

2.6 Grammar induction

Grammar inference is a special case of inductive learning, in which the goal is to create a formal grammar from positive and negative examples of the words of the language. Positive examples should belong to the language generated by the grammar, while negative should not be found in it.

There is always an infinite number of grammars satisfying this criterion; we usually look for the "simplest" grammar (e.g. with the lowest number of rules). This corresponds to the Minimal Description Length principle[35]. However, as the Kolomogorovian complexity is incomputable, usually an approximation suffices.

The basic algorithm can be described as follows. *An initial grammar G_0 is guessed. Often it is useful to specify the type of grammar (1, 2 or 3), and thus place constraints on the forms of the candidate rewrite rules. In the absence of other prior information, it is traditional to make G_0 as simple as possible and gradually expand the set of productions as needed. Positive training sentences $x+$ are selected from $D+$ one by one. If $x+$ cannot be parsed by the grammar, then new rewrite rules are proposed for P. A new rule is accepted if and only if it is used for a successful parse of $x+$ and does not allow any negative samples to be parsed.* [8]

In Chapter 7, I propose a method for grammatical inference based on Myhill-Nerode equivalence.

Chapter 3

Computational Linguistics and Natural Language Processing

3.1 Linguistics and computational linguistics

Linguistics is the science of human language. It studies the language form (morphology, syntax, phonology), language meaning (semantics and pragmatics) and the language in various context, such as the language origin and history, the use of language in society, psycholinguistics and neurolinguistics, which study language in the brain; language learning and acquisition in children; and discourse analysis, which analyzes the structure of language and language coherence and cohesion.

Computational linguistics studies the language from a computational perspective. It can be divided according to the type of language studied – spoken or written; and according to the respective task: whether the language is generated or analyzed. It is closely related to the field of natural language processing, which concerns *the interactions between computers and human (natural) languages.*[24]

3.2 Some linguistics concepts

3.2.1 Grammar

The grammar refers in linguistics to any form of systematic language description. This can mean only the rules, or also the corresponding research. There is a question how different natural languages are from formal languages (e.g. the work of Noam Chomsky).

Usually the terms syntax, morphology and phonology are used for the respective, more specific, fields. In this work, we will focus on written text, so we will skip phonology, which systematically studies the use of the sound to encode meaning.

3.2.2 Syntax

Syntax focuses on principles for constructing sentences, such as word order.

3.2.3 Morphology

Morphology studies what form the words take in the sentence, the rules which influence this form, and parts of the form. It analyzes how the words are created, and how they change depending on the context in which they occur.

3.3 Different levels of analysis

A given text can be analyzed from different perspectives. A good illustration is that the mistake(s) can occur at different levels; and still the other levels can influence the perspective.

> Vamjvms sdfklj asdfljk. (nonsensical words)
>
> Crocodile chair eleven. (collection of words without meaning)
>
> This is a sentnece. (a typo, but otherwise correct)
>
> Me hunger have. (grammatically incorrect, but with sense)
>
> Colorless green ideas sleep furiously. (Chomsky [2] – nonsensical, but grammatically correct)

Five plus seven is thirteen. (pragmatically incorrect)

Bratislava is the capital of Slovakia. (correct)

Take as an example the third sentence. It contains a typo; however it is easily understood – . Moreover, in the further context, it is a correct example of a typo – therefore if I use a spellchecker to check my work, I would not want it to be corrected. Could we even imagine some spellchecker in the future which will understand this?

Similarly, the sixth sentence does not mean that the author of this work does not know elementary math. Correcting "thirteen" to "twelve" would be absurd.

3.4 Challenges of natural language processing and AI-Completeness

Many tasks in natural language processing, such as machine translation, and also other problems of Artificial Intelligence, are considered difficult and are not solved to our satisfaction. Some scientists have hypothesized, that solving these problems is equivalent to creating a Strong AI, or in another words, to make computers as intelligent as humans.

The concept was created in analogy to NP-Completeness class in computational complexity. A formalization using human-assisted Turing machine (counting the complexity of the human and of the computer part) was proposed; however many problems in AI are not yet formalized, so this formalization has only a limited use.

Chapter 4

Slovak language

Slovak language belongs to West Slavic languages. Similar to other Slavic languages, it is a fusional (inflecting) language with a rich morphology including alternations in derivational and inflectional morphology.

4.1 Lexical categories

Slovak language has 10 lexical categories:

1. nouns (podstatné mená): denote persons, animals, things
2. adjectives (prídavné mená): denote properties of persons, animals and things
3. pronouns (zámená): substitute nouns (refer to them)
4. numerals – number names (číslovky): denote count or order
5. verbs (slovesá): express action or state
6. adverbs (príslovky): modify meaning of verbs
7. prepositions (predložky): express relations between words
8. conjunctions (spojky): connect words, phrases or sentences
9. particles (častice): express relation of speaker to the statement

10. interjections (citoslovcia): express emotions or sounds

Words in the first five categories are flective (they can change according to the context). The other five categories are inflective – words does not change. Verbs are conjugated.

4.2 Declension

Declension is the inflection of nouns, adjectives, pronouns and numerals to express the grammatical categories – gender, number, case. This is usually accomplished by adding the respective suffix to the root form of the word (stem). In a noun phrase, the grammatical categories of adjective(s) and numeral(s) match that of the noun.

The Slovak language has three genders (masculine, feminine, neutral), two possible number categories (singular and plural) and six morphological (seven grammatical) cases[1]. The nouns can be divided in roughly 4 declension paradigms for each gender (12 paradigms together). Words belonging to the same paradigm have suffixes in the respective case and number. The adjectives have 4 paradigms, numerals have 4 paradigms. Pronouns usually have each its own declension forms, some are declensed like adjectives.

The authoritative end exhaustive information on Slovak declension can be found in [9].

For illustration, I have chosen to show the declension of only one phrase: *jeden pekný chlap – one nice man.* As mentioned above, gender, number and case of the three words in the phrase are the same. The declension is shown

[1]In Slovak language, the fifth case – the vocative case is almost always equivalent to the nominative form. Only a few words retained different form, and are used in religious, literary or ironic context.

chlap – chlape (man – O, man!)
chlapec – chlapče (boy – O, boy!)
človek – človeče (human – O, man!)
Boh – Bože (God – O, God!)
pán – pane (lord – O, Lord!)

Because of this, the morphological vocative case is usually considered dead. However, there is a nice linguist joke in Slovak: “Človeče, netáraj, v slovenčine vokatív nemáme!” (“O, man [in vocative], don’t blather, we don’t have vocative case in Slovak!”)

for both singular and plural[2]:

case	**singular**	**plural**
nominative	jeden pekný chlap	jedni pekní chlapi
genitive	jedného pekného chlapa	jedných pekných chlapov
dative	jednému peknému chlapovi	jedným pekným chlapom
acusative	jedného pekného chlapa	jedných pekných chlapov
vocative	jeden pekný chlape	jedni pekní chlapi
local	jednom peknom chlapovi	jedných pekných chlapoch
instrumental	jedným pekným chlapom	jednými peknými chlapmi

4.3 Conjugation

Conjugation is the inflection of the verb to express the grammatical categories – number, tense, mood, and aspect. The number of the verb and number of the agent (usually the noun or pronoun) match each other. Most of the Slovak verbs are divided into 14 conjugation paradigms, the rest of them has irregular conjugation.

For illustration, the conjugation of the verb *otvárať – to open* in the present continuous tense (indicative mood, imperfective aspect) is shown:

person	**singular**	**plural**
first	otváram	otvárame
secong	otváraš	otvárate
third	otvára	otvárajú

4.4 Syntax

As was already mentioned, the grammatical categories of adjective(s) and numeral(s) match grammatical categories of the noun they refer to and usualy precede it. The verb matches the subject in the number.

The word order is relatively free, because the role of the word can be deduced from the rich morphology. The neutral (unmarked) word order is

[2]A reader might wonder: How a numeral "one" can possibly have a plural form? In the plural context, it is understood as a group numeral (*Jedni chlapi robili toto, druhí tamto – One group of men was doing this, and the others were doing that*), or as an emotive emphasizing (in vocative) *Vy jedni chlapi! (Ste všetci rovnakí!)* or even with the changed word order *Vy chlapi jedni!* – "oh, you men! (You are all alike!)"

Subject-Verb-Object for indicative mood and Verb-Subject-Object for interrogative mood (in question). A different word order expresses emphasis or can be used in poetic style.

Chapter 5

Machine translation

The current leading technology in machine translation is statistical machine translation. Statistical machine translation is based on statistical methods. First a number of text samples – pairs – source language/destination language is obtained. The statistics tells us what words or phrases in source language usually (with a defined probability) corresponds to what phrases in destination language, according to the Bayes theorem: $p(d|s) = p(s|d)p(d)/p(s)$.

The text we want to translate is analyzed and the phrase with the best probability is used. According to the Beyes theorem, a greater weight is assigned to longer phrases. This means that it is possible to evaluate the context – if a longer phrase is available, it is used instead of a shorter phrase, which might had not fully captured the context. A word can have a different meaning when used together with different words. This is common – a good example are the auxiliary verbs – "have" in "have been" has a different meaning from "have" in "have got".

However, even in the idealized model, a number of assumptions and simplifications must be made to make the model computationally feasible (for example, selecting a good word order is an NP-complete problem [19]).

5.1 An example – Google Translate

Perhaps the most common tool nowadays is Google Translate which we will use to illustrate statistical machine translation[16]. I will present a sample of text in English, then its automatic translation in Slovak. Next comes the correct translation in Slovak, which I have created myself by hand, to illustrate the difference. A final sample is the translation of the correct Slovak back into English using Google Translate.

1 – English source
The current leading technology in machine translation is statistical machine translation. Statistical machine translation is based on statistical methods. First a number of text samples – pairs – source language/destination language is obtained. The statistics tells us what words or phrases in source language usually (with a defined probability) corresponds to what phrases in destination language. The text we want to translate is analyzed and if a phrase occurring in the sample is found (matched), its corresponding translation is used.

2 – Translation to Slovak using Google Translator
Aktuálne predné technológie vo strojovom preklade je štatistický strojový preklad. Štatistický strojový preklad je založená na štatistických metódach. Prvé číslo textu vzoriek - dvojica - zdrojovom jazyku / určenia jazyka je získať. Štatistika nám hovorí, čo slová alebo frázy v zdrojovom jazyku sa zvyčajne (s definovanou pravdepodobnosťou) zodpovedá tomu, čo fráz v cieľovom jazyku. Text chceme prekladať je analyzovaná a ak frázy vyskytujúce sa vo vzorke nachádza (uzavreté), jeho zodpovedajúce preklad je používaný.

3 – Translation to Slovak (human)
Aktuálnou vedúcou technológiou v strojovom preklade je štatistický strojový preklad. Štatistický strojový preklad je založený na štatistických metódach. Najprv sa získa množstvo textových vzoriek – párov – zdrojový jazyk / cieľový jazyk. Štatistika nám povie, ktoré slová alebo frázy v zdrojovom jazyku zvyčajne (s určenou pravdepodobnosťou) zodpovedajú ktorým frázam v cieľovom jazyku. Text, ktorý chceme preložiť, sa analyzuje a ak sa nájde fráza vyskytujúca sa vo vzorkách, je použitý jej zodpovedajúci preklad.

4 – Translation of Slovak text(3) back into English via Google Translator

> Current leader in machine translation technology is a statistical machine translation. Statistical machine translation is based on statistical methods. First, get a lot of text samples - pairs - source language / target language. Statistics tell us that words or phrases in the source language usually (with a specified probability) which correspond to phrases in the target language. Text, we want to translate, analyze, and if found phrase occurring in samples, used its corresponding translation.

We can count the differences between samples 2-3 and 1-4 to evaluate the quality of the translation. But the reader can easily see the difference – either in 1-4 if you are a native English speaker, or 2-3 if you are a native Slovak speaker.

How it is possible to improve the translation? Let us concern with Slovak language and possibly other Slavic, inflecting languages in the same family. We see that the main source of automatic translation errors is the inflection. Introducing a grammar module could greatly improve the quality of machine translation.

5.2 Grammatically aware statistical machine translation

I think it would be possible to modify the corpus of the sample text pairs by replacing every word (both in the source and in the destination language) by its root and add the respective grammar tags. The algorithm would then have a greater possibility to match the grammatically neutral sentence; and it could take the grammar tags into account when counting the probabilities. Then in the output text generation, the respective grammar tags of the paired text would be used to create the correct grammar forms of the destination language. The separation of the roots and the grammar categories would virtually multiply the size of the corpus by the number of possible grammatical categories.

Let us illustrate this on a small example. Suppose the original corpus contains this sentence pair:

I have a child. – Mám dieťa.

This can be modified as:

(Person – first person) (to have – first person) (article – indefinite) (child). (mať – first person) (dieťa – accusative).

Now if we want to translate the sentence *You have a child*, the original method would (with a limited corpus) get us something like **Ty mám dieťa*, because it could not find the exact match. The modified method can search for the

(Person – second person) (to have – second person) (article – indefinite) (child)

pattern, and use the respective word roots, but with the correct grammatical categories (matching separately the verb "to have" in English with "mať" in Slovak, and match the tag "verb in second person" in English with "verb second person" in Slovak).

The resulting translation would be (mať – second person) (dieťa – accusative).

This will be rewritten as *Máš dieťa* what actually is a correct translation.

Chapter 6

Other Applications of Natural Language Processing

6.1 Chatterbots

6.1.1 ELIZA

ELIZA was a first chatterbot program, developed in 1966 by Joseph Weizenbaum. [37]. Its purpose was to illustrate the possibility of communication between a human and a computer in a natural language.

The name ELIZA was based on the play Pygmalion by George Bernard Shaw, because similarly to the main character of the play Eliza. As Eliza was taught to speak only on a limited number of themes, ELIZA is able to talk only in limited sense: to use catchwords and a few rules to manipulate pronouns (transform the sentence from the first person to the second person) or to use a general response like "Can you elaborate on that?". ELIZA was acting like a psychotherapist, so the patients were more confident and this response did not make them suspicious.

6.1.2 Chatterbots and Turing test

Eliza has shown that a simple program, which does not even understand the language, can "fool" human to think they speak to a human person. This is related to Turing test [36], which states that a machine, which can imitate human behavior, has to be considered intelligent.

However, if an informed person talks to a chatterbot program, it can easily detect the mechanic behavior. A chatterbot competition is held every year to evaluate the quality of the current programs. In principle, they still use the same tricks as the original ELIZA program: transformation rules (although more elaborate), general responses, and maybe a limited context preservation.

6.1.3 Catty v3 - artificial futility

An interesting attempt is Catty by Michal Zalewski. Catty does not use standard methods and even does not attempt to analyze the statements. It uses Google Search to find a fitting answer to the statement. *The exact sentence to be used is found by applying a trivial, language-blind word matching algorithm.*[38] Therefore, it can give a surprising response and answer any theme – sometimes also independently on language.

> **You:** hi, how are you?
> **Cat:** Good morning, good morning, good morning to you, good morning good morning and how do you do.
>
> **You:** What do you do today?
> **Cat:** Imagine what you could do with AAA at 350 MHz, with current Anti Alias, etc., etc.
>
> **You:** What is AAA?
> **Cat:** The author of this letter asked whether today's rich in beggar's clothes were likewise making a public statement about what sorry folks they were inside.
>
> **You:** Koľko je hodín?
> **Cat:** Sice toho za tu půl hodinu, co jsem na něj koukal skrz okno Windows Media Playeru a přes jeho webovou kameru, moc nedělal, ale je to fakt zajimavý kouknout se na někoho, jak normálně žije..

The last example question is in Slovak, the answer is in Czech[1].

[1] Czech is a close language to Slovak

6.2 Anti-plagiarism software

A common task nowadays is to detect student plagiarism. This usually requires matching student's text with the database (or Internet). However, the result is unreliable and always a human must judge the output. Students can evade detection by rephrasing the text, for example using synonyms or changing word order. On the other hand, some legitimate sentences or phrases are so common, that they are sometimes detected as plagiates (consider for example mathematical definitions).

6.3 Mechanical rephrasing

An interesting work by Páleš [33] focuses on rephrasing the Slovak sentences. Considering the rich morphological features of Slovak language, the work is exceptional in a sense. However, the code or an executable program is not available, so the results were not reproduced by another researchers. Let us illustrate using examples from [33] what are the abilities of the program:

> Lúče slnka vysušili vodu v jazere. (original sentence) – The sunrays have dried the water in the lake.
>
> Vodu v jazere vysušili lúče slnka. – The sunrays were what has dried the water in the lake.
>
> Slnečné lúče vysušili jazernú vodu. – The sunrays have dried the lake water.
>
> Voda v jazere vyschla kvôli slnečným lúčom. – The water in lake dried out because of sunrays.
>
> Lúče slnka spôsobili vyschnutie jazernej vody – The sunrays have caused the lake water to dry out.
>
> Voda jazera sa vysušila pod lúčmi slnka. – The water of the lake has dried out under the rays of sun.

6.3.1 Rephrasing as a circumvention of plagiarism detection?

A student could use such tool to automatically rephrase the text. This would circumvent many (if not all) current plagiarism detection tools. Those are based on simple comparision of strings – sentence parts, and are set up so they detect blocks of text longer than (say) 5 words, to avoid unnecessary matches. Rephrasing would reorder words and break the sentence in many chunks, which would render exact matching impossible.

6.4 Semantic text comparision

How would it be possible to match two sentences having the same (or similar) meaning? A simple approach is first to handle synonyms and disregard the word order. For synonyms, a thesaurus dictionary or a measure like Normalized Google Distance [5] could be used.

An advanced technique like Latent semantic analysis [11] uses a matrix of term occurences in documents to evaluate the similarity of the documents.

6.4.1 An echoic semantic context network

This method is inspired by the notion of brain/though association processes under the view of associativism. When reading a text, a flow of associations and images occurs in the brain. The current state of science is unable to fully simulate or even understand or describe the processes running in the brain.

However, we can approximate very roughly the associations between words and their meanings by a semantic network. An associative semantic network represents words, represented as nodes in a graph, connected according to their meaning – the more similar is the meaning of the words, the highest weight has the edge between the respective nodes. Suppose we could construct this network using the Normalized Google Distance.

During processing of the text, every time a words is read, the respective node is activated. The activation spreads also among neighboring nodes. The activation is time-dependent and deceases with the progress of time. (Compare this to interaction between Barsalou's Language System and Simulation System, as described in [1] especially Figure 13.3). The activations are not reset after the sentence; this enables to refine the meaning of homonymes in the next sentence.

The temporal profile of the nodes' activations represents the "meaning" of the text.

6.4.2 Text comparison

Now, if we want to compare two texts according to their meaning (in a limited sense) we can compare the respective temporal profiles. Let's illustrate this on a pair of sentences.
"I am hungry"
"I want to eat."

Suppose that we have previously constructed an associative semantic network consisting of words *I, am, to, eat, food, hunger, desire, want* (and others) – see Figure **??**. The meaning of the word *to* is too vague; as it servers as an preposition, it is probably connected to a lot of words. The word *am* is strongly connected to *I* and these two words do not have connection to the words *eat, food, hunger, desire*, and *want*.

When reading the first sentence, the node for *I* is activated (see Figure **??**. Because it is strongly connected to the word *am*, this node is also activated. Reading the next word *am* only strengthens this activation. Last word, *hungry*, is connected to *hunger* very strongly, so the *hunger* is activated and the activation spreads also to the nodes *desire* and *want*, and slightly also to the nodes *food* and *eat*.

When reading the second, the nodes for *I* and *am* are activated similarly as when reading the first sentence. Reading the next word *want* activates also *desire*, however the direction *desire*→*hunger* is weak; so the node *hunger* is not yet activated. Then word *to* is activated, and other nodes slightly decay in their activity. Last word, *eat*, is connected to *hunger*, this impulse together with the weak connection from *desire* activates the node *hunger*.

By comparing the temoral profile of activations the meaning of these two sentences can be evaluated to be similar.

6.5 Physicist's Eliza

In the Bachelor work [20] which I was the advisor of, the author implemented the idea of solving simple physical tasks using simple facts extraction and

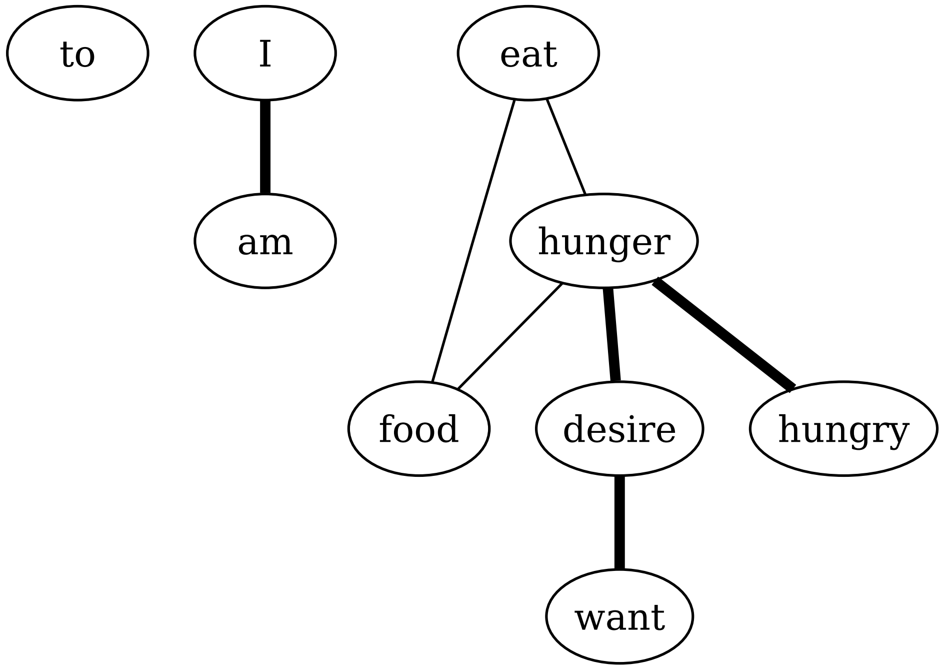

Figure 6.1: A part of the associative semantic network.

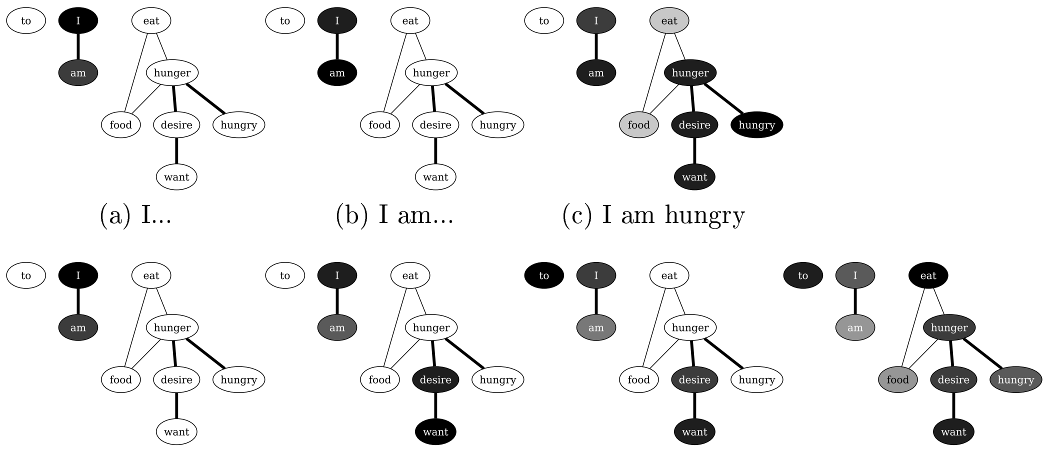

Figure 6.2: Activations in the network while reading the sentences.

feeding the facts into a database of physical formulas. The program works on tasks given in Slovak language.

As the name Eliza suggests, the program does not really have to understand physics (like the ELIZA program did not really understood the conversation). It matches the numbers and the measurement units to guess the input data, and searches for keywords to figure out the required output.

Let us suppose the problem reads [2] "Jane drives at an average speed of 50 km/h on a journey of 100 kilometers. How long does the journey take?" The program can isolate the following inputs *"50 km/h", "100 km"* and knows the required output is *"time"*. It looks up a fitting formula with variables for given data and the required output. It is then easy to evaluate the result and generate an answer according to a simple pattern "The *time* is *2 hours*".

Example 13. Solving a simple task.

U: S akým veľkým zrýchlením sa na štartovacej dráhe rozbieha lietadlo, ktoré za 20 sekúnd dosiahne rýchlosť 150 m/s?[3]

E: Čas[4]: t = 20.0 sekund (20.0 s) Rýchlosť[5]: v = 150.0 m/s (150.0 m/s)
a = v / t
a = 150.0 / 20.0
a = 7.5 m*s-2

Zrýchlenie bolo 7.5 m*s-2.[6]

[2]for illustraton in English

[3]What acceleration has a plane on a runway, which in 20 seconds gains the speed of 150 m/s?

[4]Time

[5]Speed

[6]The acceleration was 7.5 ms^{-2}.

Chapter 7

Stemming via Myhill-Nerode equivalence

In this chapter I propose a novel, language-independent method for stemming. This method uses grammar induction to create a regular grammar (equivalent to a minimal deterministic final automaton) from the list of words, using Myhill-Nerode equivalence. The nonterminals in this grammar represent sets of suffixes. This method assumes that the most used sets of suffixes are those used to create inflections. The words which differ only in an "inflectional" suffix are supposed to be different variants of the same word and can be grouped together according to the same stem. This method can be used for any inflecting language. The result of this method on Slovak language is presented.

7.1 Introduction

Stemming is the process of identifying the stem of the word. For example, a word *"mesto"* ("a city" in Slovak language) becomes *"mesta"* in genitive case – "(without) a city" – and *"meste"* in prepositional case – "(about) a city". The part of the word, to which affixes are attached in order to form a grammatical variant of the word, is called the stem, e.g. *"mest-"*, and usually does not change across different grammar variants.

7.1.1 Stemming and its use

When a user wants to read an article which mentions the word “city”, she can use a search engine to find such articles. However if the article is written in an inflecting language, e.g. in Slovak, searching the text for occurrences of the string *mesto* yields only articles which mentions “a city” in the nominative case, what is unsatisfying.

An advanced search engine can therefore search not only for occurrences of *mesto* but also for occurrences of *mesta*, *meste* and other inflection variants.

To accomplish this, the search engine has to have a dictionary of variants for every possible word.

7.2 Previous work

Several methods for automatic stemming were proposed. Some methods are directly based on rules specific to the language, e.g. English ([21],[34],), Bulgarian ([26]) and other languages; or require an external rule set ([31],[30]).

Existing language-independent approaches are based on n-gram analysis ([17],[25]) and stochastic learning from training examples ([14]).

7.3 Grammatical induction

Grammatical induction (also called grammar inference) is the process of creating a formal grammar which produces a given (formal) language.

7.3.1 Formal grammar

A formal grammar is a set of rules which describe how to form strings in a formal language. Usually a formal definition is given, which consist of: a finite set of non-terminal symbols, a finite set of terminal symbols (alphabet), a finite set of production rules, and a starting symbol; in the formal notation a quadruple (N, T, P, S). See section 2.3 on the page 11.

Regular grammar

Regular grammar is a formal grammar where rules have the form $A \to b$, $A \to bC$ or $A \to \epsilon$, where A, C are non-terminal symbols, b is a terminal symbol. The symbol ε means empty string.

In the Chomsky hierarchy (see the Figure 2.1, page 12) of grammars, regular grammars are in a lowest position. They are strictly weaker than classes of grammars higher in this position.

7.4 Method

We propose a method to create a regular grammar from a list of the words belonging to a given natural language. For each nonterminal in this grammar we count the number of its uses when generating words. The terminals with a high count represent suffixes which are often used to form variations for many different stems. One may therefore assume that the suffixes common for many stems are those which characterize the inflection system of the language. Those words which derivation is different only in using those nonterminal symbols can be grouped together because they have a common stem.

7.4.1 Myhill-Nerode theorem

Myhill-Nerode theorem[27] states that the relation R defined on words u, v from the alphabet Σ by

$$u \; R \; v \iff_{def} \forall x \in \Sigma^* (ux \in L \Leftrightarrow vx \in L)$$

is a relation of equivalence. The consequence is that the language L is regular if and only if the relation R is finite, and the states of a minimal deterministic final automaton (DFA) for language L can be constructed according to equivalence classes in R.

The idea of using the Myhill-Nerode theorem for stemming purpose was not published until now to my knowledge. It is surprising, because this method is simple and relatively straightforward.

7.4.2 Algorithm

Let us consider the given list of words to be a finite (and therefore regular) language L. Then, the algorithm implementing the relation from the Myhill-Nerode theorem and creating the grammar is quite straightforward.

The numeric threshold – the minimum number of occurences to consider the nonterminal to be ““inflectional” – is entered by hand. Then we can mark nonterminals beyond the threshold and group the words which derivations differ only by marked nonterminals. The algorithm is as follows:

1. Create a list of all prefixes of all words.
2. For each prefix, create a set of suffixes that can be attached to it so we get a word in the dictionary.
3. Prefixes with the same set of suffixes belong to the same equivalence class.
4. For each equivalence class, assign a nonterminal. For each equivalence class and a suffix assign a grammatical rule.
5. Reduce each nonterminal, which contains only one rule, by this rule in every occurrence of the nonterminal. (This will reduce chains of nonterminals to fewer, more readable, string-like rules.)
6. Count the number of uses of each nonterminal.
7. Mark the nonterminals with the count greater or equal to the given threshold.
8. Generate the words from the grammar using recursion. If you reach a marked terminal, create a new group and add there all words from all children calls.
9. Output the groups.

An implementation in C++ language is available under the GPL3 license in the Appendix A. and also electronically at [22].

7.5 Results for the Slovak language

The Slovak language belongs to the family of Slavic languages. Similarly to other Slavic languages, it has a rich morphology: endings mark gender, number, and case in declension of nouns, adjectives, pronouns and numerals; endings also mark person, number, tense, mood, and aspect in conjugation of verbs.

Table 7.1: Excerpt of resulting grammar

rules for the nonterminal	#uses
$0 \rightarrow \ldots \mid mesta\ 56 \mid meste \mid mesteck\ 61 \mid mesto\ 48 \mid mestsk\ 455 \mid mestu \mid \ldots$	(start)
$3 \rightarrow i \mid \varepsilon$	111
$15 \rightarrow ho \mid j \mid \varepsilon$	179
$48 \rightarrow m \mid \varepsilon$	101
$49 \rightarrow ch \mid m \mid \varepsilon$	111
$56 \rightarrow ch \mid m\ 3 \mid \varepsilon$	121
$61 \rightarrow a \mid o \mid u$	7
$83 \rightarrow m \mid u$	35
$455 \rightarrow a \mid e\ 15 \mid i \mid o\ 83 \mid u \mid y\ 49$	2

7.5.1 Preparation of the dictionary

We have obtained a raw dictionary of all words from the Slovak National Corpus [18]. This dictionary lists each word together with the frequency distribution, automatically obtained from texts in corpus. The dictionary was not manually checked, contains misspelled and slang words, words from other languages and other artifacts like *"www"*. It also might not contain every possible form of a word.

We have decided to consider only the words with the frequency greater or equal than 1000. We also filtered out tokens which contained a symbol which is not a letter (e.g. *"2x"*).

7.5.2 Sample of created grammar

We present an excerpt of resulting grammar related to the words beginning with *"mest"*, see Table 7.1. Numbers denote nonterminals. Each nonterminal (except the starting nonterminal 0) is assigned a count of uses. We also present a visualization of this part of the minimal automaton, see Figure 7.1.

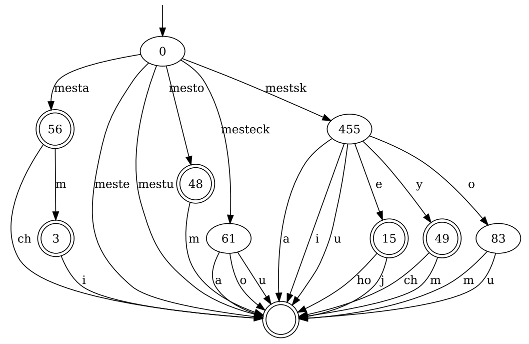

Figure 7.1: Part of the minimal automaton

7.5.3 Sample of the stemming dictionary

We present a sample of the stemming dictionary obtained by entering a threshold of 2:

- mestach mestami mestam mesta
- meste
- mestecka mestecko mestecku
- mestom mesto
- mestska mestskeho mestskej mestske mestski mestskom mestskou mestsku mestskych mestskym mestsky

7.5.4 Summary of the results for Slovak language

The results are not perfect. This is partly because inflection suffixes mix here with derivational suffixes: The words *"mesto"* ("a city"), *"mestecko"* ("little city") and *"mestsky"* ("related to city") have a slightly different meaning; and each of them occurs in several of its grammatical forms.

Let us consider an interpretation of the linguistic term "stem" where one has to distinguish between the derivational and the inflection suffixes (so called light stemming). In this interpretation, the ideal situation would be to create three stem groups:

- mestach mestami mestam mesta meste mestom mesto
- mestecka mestecko mestecku
- mestska mestskeho mestskej mestske mestski mestskom mestskou mestsku mestskych mestskym mestsky

The first group represents different forms of the word *"mesto"*, the second is for *"mestecko"* ("little city") and the third represents *"mestsky"* ("related to city").

Our method failed to identify the first group and broke it up into three subgroups. It correctly identified the second and the third group.

7.5.5 Estimating the error rate

We have manually estimated the error rate according to the methodology described in [32]. This comprises the understemming index (UI) and the overstemming index (OI).

We have manually processed the first 100 lines from the resulting dictionary. Each line consists of words with the same stem. In the ideal case, one line corresponds to one real stem – all words having this particular stem belong in one group. In the understemming case, several lines have to be merged into one group. In the overstemming case, the line has to be split into several stem groups.

The formulas for UI and OI[1] are motivated by looking at every (unordered) pair of words in a group. Each two words can be evaluated as having the same stem, or as not having the same stem. We then evaluate the number of pairs missed (for UI), or incorrectly assigned (for OI) by the algorithm in each group, and sum these numbers for all groups. We also calculate the number of pairs in correct (manual) stemming.

$$UI = \frac{\sum\limits_{\text{stem group } g} \sum\limits_{\text{line } l \text{ in group } g} 0.5 n_l * (N_g - n_l)}{\sum\limits_{\text{stem group } g} 0.5 N_g (N_g - 1)},$$

where g represents a correct group, consisting of some line(s), N_g is the number of words in this group, and n_l is the number of words in the line l. The outer sums run for each stemming group, and inner sum runs for each

[1] $OI(L)$ in Paice's notation

line which belongs to one group. Only groups consisting of more than one word are considered.

$$OI = \frac{\sum\limits_{\text{line } l} \sum\limits_{\text{stem } s \text{ in line } l} 0.5 n_s * (N_l - n_s)}{\sum\limits_{\text{line } l} 0.5 N_l (N_l - 1)},$$

where l represents a line from the dictionary, consisting of (one or multiple) real stems, N_s is the number of words in this line, and n_s is the number of words for the stem s. The outer sums run for each line, and inner sum runs for each stem, which is present in that particular line. Only lines consisting of more than one word are considered.

Understemming

The UI for our method is 0.57. This is higher compared to Paice's results for English stemmers (0.11–0.37 for three different stemmers and three different word sources[2]).

Overstemming

In the sample considered, we have seen only one case of overstemming. Therefore the OI is very close to zero (0.007), which is lower than in Paice's results (0.07–0.37).

7.5.6 Discussion

Our method is usable for light stemming. The understemming ratio is relatively high, the overstemming is almost non-existent. Our method is unable to join cases, where the stem itself changes in different grammatical forms, e.g. *mesto* (a city) – *miest* ((without) cities).

It is also worth noting that the Slovak language has much more complicated morphology than English, and also that our method does not use any specific language rules (contrary to the stemmers used in Paice's test).

[2]for the light stemming level, in [32] so called "tight groupings"

7.6 Expanding the method to prefixes and suffixes

Because our method is based on a regular grammar, it is impossible to use it directly for isolating the morphological root of the word (the part of the word which are prefixes and suffixes attached to). However, it is possible to reverse the direction in which the method reads and processes words. Then only prefixes (instead of suffixes) will be processed. It is also possible to join this information with the information obtained in a direct (non-reversed) run.[3]

7.6.1 Prefix (reverse) run

This can be done easily by reversing each word in the original dictionary, running the unmodified algorithm, and reversing each word in the resulting dictionary.

This run produces group of words having a common morphological stem and having the same grammatical ending (because the words were grouped only disregarding the prefixes). It should be noted that in Slovak language, adding a prefix can substantially change the semantics of the word[4]. Although the words could be still considered having a common morphological root, this is sometimes undesirable in practical applications.

Another interesting case is the negation, which is created by adding the prefix *ne-*. This run successfully joined the positive and negative forms.

7.6.2 Joining the information from direct and reverse run

We can use the information contained in the dictionaries created separately in the direct and in the reverse run. The words in first dictionary are grouped disregarding their inflectional suffixes; the words in the second dictionary are grouped according to one grammatical variation of a common morphological root.

[3]I wish to thank Andrej Lúčny for this idea.

[4]e.g. *tvrdenie* (a claim) – *potvrdenie* (an acknowledgment), *činný* (active) – *účinný* (effective), *držať* (to hold) – *vydržať* (to endure) – *zadržať* (to detain).

Now we can create a graph, each word being a node. We add an edge between those words, which occur in the same group (at the same line) in either of the dictionaries. The connected components in this graph are the resulting stemming groups. One component contains words which share the common morphological root, without regard to the attached prefixes or suffixes.

During this join, the errors from both runs are accumulated: if the direct run failed to merge different grammatical variations, these will remain separate also in the merged dictionary. A similar argument holds also for overstemming.

7.6.3 Results for bi-directional approach

A good example of the resulting components could be the groups containing the forms with the morphological roots *môž-*, which is common for the words *môže-nemôže* (can-cannot), the root *možn-* for words *možné, nemožné, možnosť, možno* (possible, impossible, possibility, (it is) possible).

- nemôže nemôžeme môžeme môžem môžete nemôžete môžeš nemôžeš môže nemôžem nemôžu
- nemožné možné možného nemožnosť možnosť možnostiach možnostiam možnosti možností možnosťami možnosťou nemožno možno

We have not estimated the error rate for this bi-directional run; mainly because it is a laborious manual task. It is dependent on the error rate of separate runs in both directions, which is relatively high. The bi-directional run is shown only as an illustration for further possibilities – an improvement of one-directional run will result also in improvement of the bi-directional method.

7.7 Further work

It should be possible to extend our approach using a context-free grammar induction, instead of simple Myhill-Nerode equivalence. An algorithm for context-free grammars is described for instance in [7]. Context-free grammar should perform better in identifying morphological roots and also in separating multiple suffixes or prefixes.

The stem-changes are a problematic issue. Assuming that there are regularities in the language in stem-changes, it should be possible to detect them and incorporate this information into the grammar. However, we hypothesize that (at least in Slovak language) it would be necessary to use a context-sensitive grammar, because the rules for stem-changes are mostly context sensitive (deleting or changing a part of the stem depends usually on the part itself, on the grammatical case and the paradigm, and on the preceding and the following characters)[5].

7.8 Conclusion

Our method is able to create a stemming dictionary using a regular grammar. It can isolate frequent suffixes in an inflecting language and create stemming groups accordingly. However it requires a correct and complete dictionary, because every form of a word is taken into account with equal weight (frequencies are not used). When the derivation and the inflection suffixes mix together, our method tends to distinguish between derivation and inflection.

It is also possible to use our method in a reverse manner, for identifying morphological prefixes, and subsequently it is possible to join both directions to obtain morphological (heavy) stemming. This method is only preliminar and is influenced by the errors from the one-directional runs.

The resulting dictionary has to be manually checked. The estimated understemming index for light stemming is approximately 0.57 and the overstemming index is approximately 0.007.

[5]See, for example, rules for genitive case of plural nouns at page 103 in [9]

Appendix A

Code for Myhill-Nerode Equivalence

```
/*  Creation of equivalence classes of relation R_L according to
    Myhill-Nerode theorem
    (c) Michal Maly, 2010

    uRv  <=>_def  \forall x \in \Sigma^* (ux \in L <=> vx \in L)
 */

/*
     This program is free software: you can redistribute it and/or
         modify it under the terms of the GNU General Public
        License as published by     the Free Software Foundation,
         either version 3 of the License, or (at your option) any
         later version.

     This program is distributed in the hope that it will be
        useful, but WITHOUT ANY WARRANTY; without even the
        implied warranty of     MERCHANTABILITY or FITNESS FOR A
        PARTICULAR PURPOSE. See the     GNU General Public
        License for more details.

     You should have received a copy of the GNU General Public
        License along with this program, see file COPYING.
  */

#include<iostream>
#include<string>
#include<vector>
```

```
#include<map>
#include<set>
#include<utility>
#include<algorithm>
#include<iterator>

using namespace std;
class Prefix;
class Nonterminal;

set < string > dictionary;
vector < set < string > >equivalence_classes;
vector < Prefix > prefixes;

class Prefix {
 public:
    string word;
 private:
    set < string > suffixes;
 public:
    Prefix() {
    } Prefix(const string & __word):word(__word) {
    }
    Prefix(const Prefix & prefix):word(prefix.word), suffixes(
        prefix.suffixes) {
    }

    friend bool operator <(const Prefix & a, const Prefix & b);
    friend bool operator ==(const Prefix & a, const Prefix & b);
    friend bool by_equivalence_class(const Prefix & a, const
        Prefix & b);

    void compute_suffixes() {
        int word_size = word.size();
        for (set < string >::iterator dictionaryWord = dictionary
            .lower_bound(word); dictionaryWord != dictionary.end
            (); ++dictionaryWord) {
            if (string(*dictionaryWord, 0, word_size) == word)
                suffixes.insert(string(*dictionaryWord, word_size
                    ));
            else
                break;
        }
    }

```

```
};

void compute_suffixes(Prefix & k)
{
    k.compute_suffixes();
}

bool operator <(const Prefix & a, const Prefix & b)
{
    return a.word < b.word;
}

bool operator ==(const Prefix & a, const Prefix & b)
{
    return a.word == b.word;
}

ostream & operator <<(ostream & o, const Prefix & k)
{
    return o << k.word;
}

bool by_equivalence_class(const Prefix & a, const Prefix & b)
{
    return a.suffixes < b.suffixes;
}

void add_prefixes(const string & word)
{
    //allow also the empty word
    for (unsigned int i = 0; i <= word.size(); i++) {
        prefixes.push_back(Prefix(string(word, 0, i)));
    }
}

map < string , int >table;

int which_class(const string & word)
{
    map < string , int >::iterator which = table.find(word);
    if (which == table.end()) {
        /*  in the case we find no class , we return the  class
            that which all other words are in */
        return equivalence_classes.size();
    } else {
        return which->second;
    }

}

typedef pair < string, Nonterminal * >Rule;

int nonterminalNumbers = 0;

class Nonterminal {
 public:
    int symbolNumber;
     vector < Rule > rule;
    bool modified;
    bool outputted;
    int usesCount;
 public:
  Nonterminal(int x = -1):symbolNumber(x), modified(false),
      outputted(false), usesCount(0) {
    } void modify_rules() {
        if (modified)
            return;
        modified = true;

        for (unsigned int i = 0; i < rule.size(); i++) {
            Rule oldrule = rule[i];
            Nonterminal *xi = oldrule.second;

            if (xi)
                xi->modify_rules();

            /*  if the derivation is clear or the nonterminal is
                useless, we rewrite it */

            if (xi && xi->rule.size() == 1) {
                Rule newRule = Rule(oldrule.first + xi->rule[0].
                    first, xi->rule[0].second);
                rule[i] = newRule;
                continue;
            }

            if (xi && xi->usesCount == 1) {
                //delete the old rule
                rule.erase(rule.begin() + i);
                //...and replace by a series of new rules
                for (int j = 0; j < xi->rule.size(); j++)
                    rule.insert(rule.begin() + i, Rule(oldrule.first + xi->rule[j].first, xi->rule[j].second));
            }

        }
    }
    void makeSymbolNumbers() {
        if (symbolNumber != -1)
            return;
        symbolNumber = nonterminalNumbers++;

        for (unsigned int i = 0; i < rule.size(); i++)
            if (rule[i].second)
                rule[i].second->makeSymbolNumbers();
    }

    void output() {
        if (outputted)
            return;
        outputted = true;

        cout << usesCount << '\t' << symbolNumber << " -> ";
        for (unsigned int i = 0; i < rule.size(); i++) {
            if (i)
                cout << " | ";

            if (rule[i].second)
                cout << rule[i].first << " " << rule[i].second->symbolNumber;
            else {
                if (rule[i].first != "")
                    cout << rule[i].first;
                else
                    cout << "\\eps";
            }
        }
        cout << endl;

        for (unsigned int i = 0; i < rule.size(); i++)
            if (rule[i].second)
                rule[i].second->output();
    }

    void generate(string s, bool grouping) {
        bool splitgrouphere = false;
        if (usesCount >= 2 && grouping) {
            splitgrouphere = true;
            grouping = false;
        }

        for (unsigned int i = 0; i < rule.size(); i++) {

            string sExt = s + rule[i].first;

            if (rule[i].second) {
                rule[i].second->generate(sExt, grouping);
            } else {
                cout << sExt;
                if (grouping)
                    cout << endl;
                else
                    cout << " ";
            }
        }

        if (splitgrouphere)
            cout << endl;

    }
};

Rule modify(Rule rule)
{
    Nonterminal *xi = rule.second;
    if (xi)
        xi->modify_rules();

    /* if the derivation is clear or the nonterminal is useless,
       we rewrite it */
    if (xi && (xi->rule.size() == 1 || xi->usesCount == 1)) {
        Rule wasModified = modify(xi->rule[0]);
        return make_pair(rule.first + wasModified.first,
            wasModified.second);
    } else
        return rule;
}

int main()
```

```
{
    //load the words
    while (cin) {
        string word;
        cin >> word;
        dictionary.insert(word);
    }
    cerr << "count of (unique) words = " << dictionary.size() <<
        endl;

/*  we create all possible strings -- prefixes of the words. If
    we attach something to it, we have a chance to get a word
    from the language L*/
    for_each(dictionary.begin(), dictionary.end(), add_prefixes);
    cerr << "number of prefixes = " << prefixes.size() << endl;
    sort(prefixes.begin(), prefixes.end());
    vector < Prefix >::iterator last = unique(prefixes.begin(),
        prefixes.end());
    prefixes.resize(last - prefixes.begin());
    cerr << "total number of unique prefixes = " << prefixes.size
        () << endl;

/*  compute the set of allowable suffix for every prefix*/
    cerr << "computing suffixes..." << endl;
    for_each(prefixes.begin(), prefixes.end(), compute_suffixes);
/*  sort by the equivalence class -- prefixes belonging to the
    same class are together in a row*/
    cerr << "sorting..." << endl;
    sort(prefixes.begin(), prefixes.end(), by_equivalence_class);
    cerr << "constructing equivalence classes..." << endl;

/*  create the equivalence classes */
    set < string > equivalenceClass;
    for (vector < Prefix >::iterator prefix = prefixes.begin();
        prefix != prefixes.end(); ++prefix) {
        equivalenceClass.insert(prefix->word);
        table.insert(pair < string, int >(prefix->word,
            equivalence_classes.size()));

        if ((prefix + 1) == prefixes.end() ||
            by_equivalence_class(*prefix, *(prefix + 1))) {
            equivalence_classes.push_back(equivalenceClass);

            equivalenceClass.clear();
        }
    }
    cerr << "number of equivalence classes = " <<
        equivalence_classes.size() << endl;

    //free some memory
    prefixes.clear();

/*  according to the constructed equivalence classes it is
    possible to create a finite state automaton which accepts the
    given language, where

    [x] is the equivalence class, which contains the word  x
    K = {[x] | x \in \Sigma^*}
    \delta([x],a) = [xa]
    F = {[x] | x \in L}
    q0=[\eps]
 */

/*  and also to create a grammar G=(N,T,P,\Sigma) which generates
    the given language, where

    N = K
    T = \Sigma
    P = {q -> ap   | \delta(q,a) = p} \cup
        {q -> \eps | q \in F}
 */

/*  we have one nonterminal to every class;
    except we do not have a class for the other words (for which
        we cannot generate anything)
*/
    vector < Nonterminal > nonterminal(equivalence_classes.size()
        + 1);

    cerr << "constructing nonterminals..." << endl;
    for (unsigned int i = 0; i < equivalence_classes.size(); i++)
        {
        cerr << i << "\r";
        string word = *equivalence_classes[i].begin();
        char character[2];
        character[1] = 0;
        for (character[0] = 1; (unsigned char)(character[0]) <
            255; character[0]++) {
            int equivalenceClass = which_class(word + character);
            if ((unsigned int)equivalenceClass !=
                equivalence_classes.size()) {
                nonterminal[i].rule.push_back(make_pair(string(
                    character), &nonterminal[equivalenceClass]));
                nonterminal[equivalenceClass].usesCount++;
            }
        }

        //if it is a final state, add rule for \eps
        if (dictionary.count(word) != 0)
            nonterminal[i].rule.push_back(Rule(string(""), 0));
    }

/*  to save some nonterminals:
    if the nonterminal contains only one rule q->ap,
    we follow that rule to the point we find a nonterminal

    q-a_0p_0
    p_0->a_1p_1
    ...
    p_n->a_{n+1}p{n+1} | b_{n+1}r{n+1}

    containing more rules. We change the rule to q->a_0a_1...a_n
        p_n

    We throw away unused nonterminals (we don't output them).
*/

    //begin with the [\eps] class
    cerr << "[\\eps] = " << which_class("") << endl;
    cerr << "reducing the grammar..." << endl;
    nonterminal[which_class("")].modify_rules();

    cerr << "numbering the nonterminals..." << endl;
    nonterminal[which_class("")].makeSymbolNumbers();

    cerr << "number of reduced nonterminals = " <<
        nonterminalNumbers << endl;

    cerr << "outputting..." << endl;
    nonterminal[which_class("")].generate("", true);
    return 0;
}
```

myhill_nerode.cpp

List of Figures

Bibliography

[1] L. Barsalou, A. Santos, K. Simmons, and C. Wilson. Language and simulation in conceptual processing. *Symbols and embodiment: Debates on meaning and cognition*, pages 245–283, 2008.

[2] N. Chomsky. *Syntactic structures.* The Hague/Paris: Mouton & Co., 1957.

[3] N. Chomsky. *Aspects of the Theory of Syntax.* The MIT Press, 1965.

[4] N. Chomsky. *A minimalist program for linguistic theory.* Distributed by MIT Working Papers in Linguistics, 1992.

[5] R.L. Cilibrasi et al. The Google Similarity Distance. *IEEE Transactions on knowledge and data engineering*, pages 370–383, 2007.

[6] Wikipedia contributors. Language families.jpg, 2010. [Online; accessed 12-Jan-2011] `http://upload.wikimedia.org/wikipedia/commons/a/a8/Language_Families.jpg`.

[7] Crespi-ReghizziStefano. An Effective Model for Grammar Interference. In *IFIP Congress (1)*, pages 524–529, 1971.

[8] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification*, volume 2. Citeseer, 2001.

[9] Ladislav Dvonč, Gejza Horák, František Miko, Jozef Mistrík, Ján Oravec, Jozef Ružička, and Milan Urbančok. *Morfológia slovenskeho jazyka [Morphology of the Slovak Language].* Vydavateľstvo Slovenskej Akademie Vied, Bratislava, 1966.

[10] J.L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[11] Deerwester S. et al. Computer information retrieval using latent semantic structure. U.S. Patent #4,839,853 issued June 13, 1989, filed 1988.

[12] I. Farkaš and M.W. Crocker. Syntactic systematicity in sentence processing with a recurrent self-organizing network. *Neurocomputing*, 71(7-9):1172–1179, 2008.

[13] Igor Farkaš and Ján Švantner. Učenie nesusedných závislostí pomocou rekurentných neurónových sietí. In *Kognice a umělý život VII*, pages 389–394. Opava: Slezská univerzita, 2007.

[14] John A. Goldsmith, Derrick Higgins, and Svetlana Soglasnova. Automatic Language-Specific Stemming in Information Retrieval. *Lecture Notes In Computer Science; Vol. 2069*, 2000.

[15] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to automata theory, languages, and computation*, volume 3. Addison-wesley Reading, MA, 1979.

[16] Google Inc. Inside Google Translate. Available at: `http://translate.google.com/about/intl/en_ALL/`, 2011.

[17] Anni Järvelin, Antti Järvelin, and Kalervo Järvelin. s-grams: Defining generalized n-grams for information retrieval. *Information Processing and Management: an International Journal*, 43(4), 2007.

[18] Jazykovedný ústav Ľ. Štúra SAV. Frequency Statistics for the prim-4.0, Slovak National Corpus. Available at: `http://korpus.juls.savba.sk/stats/prim-4.0/word-lemma/prim-4.0-juls-all-word-frequency.txt.gz`, 2009.

[19] K. Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615, 1999.

[20] Miroslav Kocun. Fyzikálna eliza, 2010. Bachelor's Thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava.

[21] J B Lovins. Developing of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11(1), 1968.

[22] Michal Malý. Source code for Myhill-Nerode equivalence stemming. Available at: `http://mmm.ii.fmph.uniba.sk/myhill_nerode.zip`, 2010.

[23] D.R. Marlow and M. FLOOK. Textbook of Pediatric Nursing. *AJN The American Journal of Nursing*, 70(1):146, 1970.

[24] D. McDermott and E. Charniak. Introduction to Artificial Intelligence, 1985.

[25] Paul Mcnamee and James Mayfield. N-Gram Morphemes for Retrieval. 2007.

[26] Preslav Nakov. Building an inflectional stemmer for Bulgarian. *International Conference Computer Systems and Technologies*, 2003.

[27] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

[28] E.L. Newport and R.N. Aslin. Learning at a distance I. Statistical learning of non-adjacent dependencies. *Cognitive psychology*, 48(2):127–162, 2004.

[29] C. Noam. Three models for the description of language. *IRE Transactions on Information Theory*, page 113124, 1956.

[30] C.D. Paice and M. Oakes. A concept-based method for automatic abstracting. *Library and Information Commission*, 1999.

[31] Chris D. Paice. Another stemmer. *ACM SIGIR Forum*, 24(3):56–61, November 1990.

[32] Chris D. Paice. Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.

[33] E. Páleš. SAPFO – Parafrázovač slovenčiny. *Veda vydavatel'stvo SAV*, 1994.

[34] M F Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[35] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[36] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[37] J. Weizenbaum. ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966.

[38] Michal Zalewski. Catty v3 - artificial futility (readme). Available at: `http://lcamtuf.coredump.cx/c3/README`, 2004.

[39] Miroslav Ľos. Akvizícia gramatiky z korpusu pomocou samoogranizácie, 2007. Master's Thesis, Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava.