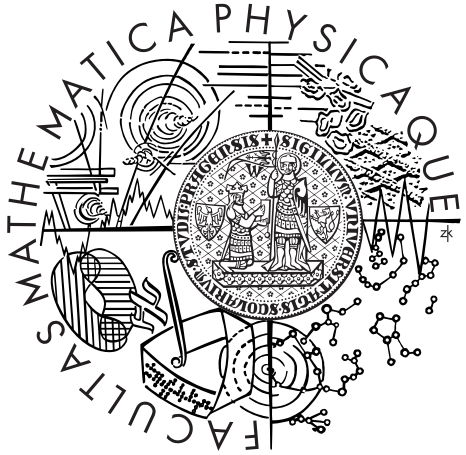
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Karel Bílek

A Comparison of Methods of Czech-to-Russian Machine Translation

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	doc. RNDr. Vladislav Kuboň, Ph.D.
Study programme:	Informatika ????
Specialization:	Asi něco s lingvistikou

Prague 2014

Dedication. I love everybody. (TODO)

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In …….. date …….. signature of the author

Název práce: Porovnáni metod česko-ruského automatického překladu

Autor: Karel Bílek

Katedra: Ústav formální a aplikované lingvistiky

Vedoucí diplomové práce: doc. RNDr. Vladislav Kuboň, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Abstrakt abstrakt.

Klíčová slova: Čeština, ruština, strojový překlad

Title: A Comparison of Methods of Czech-to-Russian Machine Translation

Author: Karel Bílek

Department: Institute of Formal and Applied Linguistics

Supervisor: doc. RNDr. Vladislav Kuboň, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Abstrakt abstrakt.

Keywords: Czech, Russian, Machine translation

Contents

Introduction

Czech and Russian are languages with a long common history.

They both belong to a Slavic family of languages, together with 10 other languages[1], which tie them together from approximately 2000 BC.[2] If we consider more recent events, in the 20th century, Czechoslovakia was a part of so-called Eastern Bloc and Soviet Union was its dominant force. Without going too far into politics, we can see why was the translation between those two languages considered important.

As for the Slavic connection – Slavic languages are usually divided into three subgroups – South Slavic, West Slavic and East Slavic. While languages inside each of those groups are in general mutually intelligible, intelligibility accross those groups is lower; still, the Slavic languages in general have very similar morphology, syntax and vocabulary.

Czech language belongs to the West Slavic group, while Russian belongs to the East Slavic group. Therefore, the intelligibility is lower – so doing machine translation makes sense – but some aspects of the language are still similar. In the work described in this thesis, I am trying to ask if the similarities are somehow exploitable for the machine translation task.

As for the political connection – because of the political role of Russian, significant efforts were directed towards building an automatic translation system between Russian and Czech, mainly during mid-eighties with the RUSLAN system. The efforts were terminated in 1990 in the final phases of development.[3]

After the velvet revolution, political dependence on Soviet Union (and then just Russia) was weakened, while the economical and cultural dependence on English-speaking countries was strengthened. The efforts of automatic translation were then more focused on English.

In the process, general direction of machine translation efforts shifted from rule-based systems to more statistical-based systems. In this thesis, I am trying to describe this historical system and compare it with some more modern approaches.

After a brief introduction to the two languages and its similarities and differences, I will describe all the available Czech-to-Russian translation systems, the amount of work needed to get them running and the work we spent on their additional development. Then I will describe a common set of test data and the exact testing procedure and lastly, I will describe the results of the described tests.

In general, I don't have ambitions for any scientific breaktrough in the Czech-to-Russian translation; I just hope that I have successfully compiled and implemented all the existing approaches and compared their general translation quality.

[1] The final count depent on exact definition. See for example SUSSEX; CUBBERLEY, 2011 or SIEWIERSKA; UHLÍŘOVÁ, 1998 – the final language count depends on whether Upper and Lower Sorbian are taken as different languages or not; whether Kashubian is a separate language, or just a variety of Polish; whether Serbian and Croatian are treated as separate languages or not

[2] See the chapter 1.1.1.

[3] See for example BOJAR; HOMOLA; KUBOŇ, 2005, HAJIČ, 1987, OLIVA, 1989

1. Brief comparison of Czech and Russian

1.1 Slavic in general

This section is an overview of history of Slavic languages and also of some characteristics of Slavic languages in general.

If a reader is interested in a further study of Slavic languages as a whole from linguistic perspective, I can only recommend SUSSEX; CUBBERLEY, 2011. Unless otherwise noted, information in this chapter are taken from this book.

1.1.1 History

The Indo-Europeans occupied European lands from approximately 4000 BC. Their language, Proto-Indo-European (PIE), is not known and can only be linguistically reconstructed.

In approximately 2000 BC, a Proto-Slavic (PS) language started to emerge. Until approximately 400 AD, the language was fairly uniform and the land occupied by Slavs was broadly coherent, stretched approximately from Oder river (Odra in Polish) to Dnieper (Днепр in Russian).

1.1.2 Slavic languages overview

Morhpology

On the morphological typology, Slavic languages belong to synthetic inflectional languages. It's morphologically rich, with a sophisticated system of prefixes, roots and suffixes, with some analytical approaches (for example, in verb morphology).

Slavic words consist of one or more roots – the main component of the word – and then several prefixes and suffixes. Prefixes usually modify the word's meaning somehow (for example, "ne-" for negative), while suffixes are either derivational, inflectional or post-inflectional (appearing in this order).

The derivational suffixes can determine word's class with processes like nominalization, verbalization, adjectivalization. Next are inflectional suffixes (also called endings), that mark one of several categories.

to wash (someone)	mýt	мыть
to wash (self)	mýt se	мыться

Table 1: Reflectives in Russian vs. Czech

Post-inflectional suffixes are found in some languages (like Russian) for example for reflectivity, whereas in other languages a separate word is used. See for example Table 5 (also compare to the voices in the section 1.1.3).

Verbs	Nominals	Both
Tense	Case	Person
Aspect	Definiteness	Gender
Mood	Deixis	Number
Voice		

Table 2: Inflectional categories

Inflectional categories that are presented in the Slavic languages are described in the Table 2 (definiteness and deixis as inflectional categories are, however, not relevant to either Czech or Russian, but are used as such in Macedonian or Bulgarian).

Slavic languages are surprisingly uniform in the word formation system, and the operations and grammatical processes are simmilar – however, the affixes themselves are usually either wholly different, or have vastly different semantic and stylistic properties – either absolutely, or in combination with other words.

The result of the described morphological richness is that the number of word forms arise significantly, especially when compared with more analytical languages (like English). As you can see in the section XXX, in the same corpus, Czech and Russian have *significantly* more word forms, than English.

Word order

In English, syntactic relations are marked mainly by word order. In Slavic languages, however, those relations are marked instead by three types of inflexion: concord, agreement and government.

For example, subject *agrees* with predicate in number, person and gender.

Because the relations are marked by inflexion, it "allows" the language to be of freer word order; and indeed, in Slavic languages, the word order variations are more common than in other languages. The *standard* order is SVO, however, this is possible to change when different emphasis is needed.

In particular, this reffers to the so-called Functional Sentence Perspective. Very simply said, it states that sentence could be split in two parts – Topic and Comment, appearing in this order and being separated by a verb. Topic is the part of sentence, *about which something is said* – while Comment is the *new information presented*. Most importantly, Topic doesn't have to be a grammatical subject of the sentence – for example, the Czech sentence "Ve městě bydlí strašidla" (*Ghosts live in the city*, literally *In city live ghosts*).

Slavic languages usually (with some exceptions like Bulgarian and Macedonian) don't have particles or any other means of marking definitiveness – except for definite information being in the topic of the sentence. In other view, the word order and functional sentence perspective can be seen as "replacing" the definitive particles, and is usually translated as such in translation to English.

In respect to the translation between English and Slavic languages, word order can be seen as a nuisance and hard to translate correctly. However, with translation between Slavic languages, it can actually help us, since we can preserve the word order and move the words less.

1.1.3 Some changes from PIE to PS

Cases

genitive	invocant nomen **Domini** nostri Iesu Christi	who call on the name **of** our **Lord** Jesus Christ	všem, kdo na jakémkoli místě vzývají jméno našeho společného **Pána** Ježíše Krista
ablative	gratia vobis et pax a Deo Patre nostro et **Domino** Iesu Christo	Grace and peace to you **from** God our Father and **the Lord** Jesus Christ	Milost vám a pokoj od Boha, našeho Otce, a od **Pána** Ježíše Krista

Table 3: Demonstrating ablative and genitive conflation on 1 Corinthians

PIE had seven cases - nominative, accusative, genitive, dative, instrumental, locative, ablative and vocative. In PS, ablative and genitive were conflated into just genitive.

Two phrases from New Testament can offer us a demonstration.

Old Latin retained both ablative and genitive. Latin genitive of *dominus* (lord, master) is *dominī*, latin ablative of the same word is *dominō*. Both these forms were used in the very beginning of 1 Corinthians in Latina Vulgata (latin version of The Bible).

In the Table 3, I have shown the translation to both Czech and English. We can see that both cases are inflated in Czech as genitive "(našeho) pána".[1]

Numbers

singular	one cow	ena krava	jedna kráva
dual (in Slovenian)	two cows	dve kravi	dvě krávy
plural	three cows	tri krave	tři krávy

Table 4: Demonstrating duals on Slovenian

PIE had three numbers, singular, plural and dual. In PS, dual slowly disappeared, while still retaining some of its usage.

However, dual disappeared in most of Slavic languages later (including Czech and Russian), leaving only traces in the grammar. One of the languages where dual remained is Slovenian.

To illustrate, I have added a comparison of "one cow", "two cows" and "three cows" in Slovenian and Czech in Table 4.

In Czech, dual was retained in declensions of several words, like "hands"; in Russian, the dual "рукама" survives in some dialects, but is generally incorrect.[2]

Genders

PIE had three genders, masculine, feminine and neutral. PS retained these genders, adding categories Personal and Animate.

[1] Bible sources: WIKISOURCE, 2013, BIBLICA, 1973, 1978, 1984, 2011, FLEK et al., 2012. Note that English and Czech translations are not actually translations from Latin, but from more primary sources, but it will suffice for this simple comparison.

[2] See OFFORD, 1996, page 18.

Tenses

PIE had six tenses: present, future, aorist, imperfect, perfect and pluperfect.

The tenses were somehow retained in PS; however, future, perfect and pluperfect were in many cases re-formed analytically — creating more complex forms with auxiliary verb and either an infinitive or past participle – for example, "budu zpívat" (I will sing) in Czech, or "буду петь" in Russian.

Moods

PIE had four moods: indicative, subjunctive, optative and imperative. In PS, imperative forms were replaced by the optatives, and subjunctive mood slowly became conditional.

Voices

active	νιπτω	I wash (someone)	myji (někoho)
medium	νίπτομαι	I wash (myself)	myji se
passive	νίπτομαι	I am washed (by somebody)	jsem myt

Table 5: Voices in Classic Greek vs. Czech

PIE had an active and a mediopassive voice.

In PS, this was refined as a reflexive and a non-reflexive voice, with the addition of a passive voice.

To illustrate, I have added an example of Classic Greek that still retained the mediopassive voice, in Table 5.[3]

Aspects

imperfective determinate	нести́	nést
imperfective indeterminate	носи́ть	nosit
perfective	понести	ponést

Table 6: Aspects in Czech and Russian

PIE had distinction between two aspects – eventive and stative; eventive aspect being further divided into perfective and imperfective aspect.[4]

In PS, the stative aspect is degramatized[5]; however, the perfective/imperfective distinction became more important in PS than in other Indo-European languages. Also, the imperfective motion verbs were further split into determinate and non-determinate.

The determinate/non-determinate and perfective/imperfective distinction is retained in both Czech and Russian. See the Table 6 and note, how hard would be to correctly translate the distinction into English.

[3]See for example PARKER, 2009, ARCHIBALD, 2008

[4]See RINGE, 2008, page 24

[5]See ANDERSEN, 2013

1.2 Drifting of Czech and Russian

In about 5th century AD, the relative unity of Slavic started to break up, caused mainly by migration, which was partly caused by expansion to the north and east by the Eastern Slavs, partly by dissolution of Roman and Hun empires and the resulting vacuum in Central Europe.

One group of Slavs moved westwards, reaching approximately what is now Poland, Czech Republic and parts of Germany, creating the so-called West Slavic language group, which Czech language is a member of. The other group moved south to Balkan, creating a South Slavic language group, which I won't discuss further in this thesis. Eastern Slavs moved north and east, eventually creating an Eastern Slavic language group. Slavic was divided into those three groups at about 10th century AD.

The differences between the languages themselves are actually less syntactical and are more phonological, morphonological and morphological, with some broad-scale lexical changes. For that reason, I am not describing the linguistic differences of the languages any further, since it would be, again, not in a scope for this thesis. Again, I can recommend the book SUSSEX; CUBBERLEY, 2011 for anybody interested in the deeper differences.

2. Translation systems

2.1 Statistical vs. rule-based – an overview

Machine translation (MT) systems has historically used many different approaches. One way of classifying the approaches is on the axis of rule-based vs. statistical.

In general, we can re-use the descriptions, used by BOJAR, 2012, which is as follows:

- rule-based MT systems:
  - use analysis, transfer and synthesis steps
  - use formal grammars
  - use hand-made dictionaries
  - have linguistic information hard-coded and therefore aren't language-agnostic
- statistical MT systems
  - use more variants of outputs, rank them with some score, and choose the best one
  - train internal dictionaries from big parallel data
  - have more compact translation core, their inner working are less obvious
  - use statistics instead of linguistic rules and therefore are more language-agnostic

However, with some of the modern systems, the distinction is not as clear-cut as we would like, for the purpose of our comparison. For example, statistical MT systems like Moses can get significantly better with some added linguistic information; on the other hand, systems like TectoMT, which can sometimes be classified as more rule-based, actually have big modules more or less based on statistics.

2.2 Unrunnable systems

In this section I will present some historical systems, that I haven't been able to get successfully running.

2.2.1 RUSLAN

RUSLAN is a machine-translation system, developed between 1984 and 1988 at several departments of Charles University, Prague. RUSLAN firmly belongs to the *rule-based* category, since at that timeframe, statistical machine translation wasn't even invented yet.

Q-Systems

Q-Systems (sometimes also Systems Q) – Q stands for "Quebec" – are a tool for machine translation, developed at Montreal University by Alain Colmerauer, also the creator of Prolog.[1]

Q-Systems are similarly declarative as Prolog, focusing more on the result than the procedural order of analysis. If there is any ambiguity, all the possibilities are explored *in parallel*. In theory, this could make writing the lexical rules easier and the rules themselves more readable; in reality, the rules are still quite unreadable, as will be seen later.

Q-Systems are not very widely used or widely worked with. One of the reasons might be the fact that all documentation is in French. However, NGUYEN, 2009 describes[2] modern-day experiments with Systems Q, also mentioning, that all Fortran implementations has been lost[3] and that he reimplemented it in C. I haven't been able to try this version.

RUSLAN components

Description of the system can be found in OLIVA, 1989 or HAJIČ, 1987 – however, the reader has to bear in mind that not only the systems are severly dated, but so is their manual and description.[4] Contemporary (but not as detailed) description of the system can be found in BOJAR; HOMOLA; KUBOŇ, 2005.

The whole RUSLAN system has several components:

- preprocessing, written in Pascal
- morphological analysis, using dictionary, written in Q-Systems and interpreted in Fortran IV
- syntactico-semantical analysis, using morphology, also written in Q-Systems; this component uses FGD as its theoretical starting point
- generation, also using Q-Systems
- morphological synthesis, using Pascal

RUSLAN uses a Czech-to-Russian dictionary, written by hand in afforementioned Q-Systems. Dictionary item looks like this:

```
DLOUH==M(RS(+(*INT)),MI2289,DLINNYJ).
DLOUH==M(RS(-(*INT)),MI2276,DOLGIJ).
```

This describes two possibilites of the translation of the word "dlouhý" to Russian: the first is "длинный" and the second is "долгий". They differ by the semantic feature INT they require or forbid from the word they depend on.

More complex dictionary item looks like this:

[1] See COLMERAUER, 1970.

[2] Unsurprisingly, also in French.

[3] Which would make UFAL's version, if it was working, quite unique.

[4] At least for me personally, especially the book OLIVA, 1989 was hard to read and hard to find the needed information in.

```
C3ES3TIN
  ==Z(@(*A), MIO109, $(JAZYK),
     2(POS, #($), &, $(MI28), $(C2ES2KIJ),
       1(=,@($), #($),$($))),
     1(=,@($),#($),$($))).
```

This item translates the word “čeština” to Russian words “чешский язык” and also describes their relationship.

Maybe because memory was more expensive than today, all similar rules are in the dictionary without any comments, leaving only very difficult-to-decypher rules.

The rules for analysis are even less readable. Random examplo of two such rules are as follows:

```
1(B*, X*1, /, X*2, F*1(C*), X*3, /, X*4, @(V*), X*5, %(X*),
 I(*), 1(X*6, $($)), X*7)

1Z(A*9), (Z*2)
  == 1(D*, /, X*2, F*1/X*),/,@(V*),X*5,%(X*),1*,1(X*6,$($)),X*7,
       A*B,
     5(U*1, @(U*2), U*3, $(U*), 3(E*(Y*1), B*(C*), W*1, W*3, %(X*),
         $(W*)),
       +1Z(A*9, Z*2)
         / -NON- (, + -DANS- X*9 -ET- +(V*) -HORS- X*9, +(VZT)
          -ET- -(V*) -HORS- X*9, *
          -ET- C* = S
          -ET- X*3,* -HORS- /,N(S), S(S), D(S), A(S), L(S), I(S)
          -ET- / -HORS- X*2, 2
          -ET- (, Y%1 = -NUL-
               -OU- E*(Y*1) -HORS- U*2,*
               -OU- E*(+(V*, *)) -HORS- U*2, *
               -OU- -NON- E*(-(V*)) -HORS- U*2, * .)
          -ET- (. E*(Y*1) *N
               -OU- H(B*(C*)) -DANS- U*1 .) .
```

This is all left with very little comments. For example, the only comment for the two rules above and about 20 more is `RELATIVE CLAUSES ADJOINED TO THEIR HEADS`.

Dictionary coverage of WebColl

The dictionary contains about 8,000 lexical items. However, the domain of the translation and, therefore, the dictionary, was manuals for old computers from 1980’s.

In different experiments (BÍLEK; KLYUEVA; KUBOŇ, 2013), we tried to measure how many nouns from the RUSLAN dictionary appear at all in a modern text.

For that, we used a monolingual Czech corpus WebColl, consisting of roughly 7 million sentences (114 million tokens)[5].

RUSLAN dictionary has 2,783 nouns. In the WebColl corpus, from those nouns, 611 appear less than 10 times – and from those, 412 don’t appear *at all*.

[5]See SPOUSTOVÁ; SPOUSTA; PECINA, 2010

The reverse is similarly infavourable: from 39,434,505 nouns in the corpus, only 11,862,221 is in the dictionary.

Experiments

Despite the general un-maintainability of the RUSLAN code and despite the small dictionary, we tried to run the system on our test data.

However, all our experiments ended in some sort of error.

Because I am not able to code in neither Systems-Q nor FORTRAN (in which the Systems-Q interpreter is coded), I gave up on this experiment. (?????)

2.2.2 Česílko 1.0

"Česílko" is a name for two entirely different machine translation systems with slightly different goals and, more importantly, slightly different structure. Both were originally intended for Czech-to-Slovak translation.

Česílko 1.0 was a system, developed in 2000, and was aimed for direct translation between Czech and Slovak and intended to assist a translation memory[6]. The translation works lemma-by-lemma in a following fashion:

- morphological analysis of source (Czech) language
- disambiguation
- direct translation, lemma-by-lemma
- morphological synthesis

The system is written in a mixture of C, C++ and Flex (fast lexical analyser generator). The code itself is not really well documented and modular, but that can be attributed to the age of some of the components – despite the whole system being developed in 2000, some files seem to be as old as 1991.

This system itself is not very extendable from Slovak to Russian. Partly because of the design itself, partly because – as we can see on the examples in the section **??** – the sentences are not really translatable word-by-word.

For that reason I decided not to further experiment with Česílko 1.0 for Czech-to-Russian machine translation.

2.2.3 Česílko 2.0

Česílko 2.0 is a different project with similar goals, but using different frameworks and adding more transfer rules[7]. However, it has its own shares of problems, that prevented us to use it.

The system works in a following fashion:

- **non-deterministic** morphological analysis of source Czech language
- translation of lemmas
- applying transfer rules by changing syntactic tree
- morphological synthesis

[6] See HAJIČ; HRIC; KUBOŇ, 2000.

[7] See HOMOLA; KUBOŇ; VIČIČ, 2009

- ranking of all the generated sentences

Unlike Česílko 1.0, Česílko 2.0 uses a non-deterministic parser and explores all the possibilities in parallel.

The more advanced transfer would, in an ideal world, make the system more modular and extensionable for our purposes. However, the implementation details prevented us from doing any significant work on Česílko 2.0.

To illuminate why, let me focus a little on the technical details.

Objective-C

Objective-C is a very simple and elegant extension of C language, developed by Brad Cox in 1980s by adding Smalltalk features to C[8].

Objective-C is, in my opinion, very easy to learn and understand, at least compared to C++, its more popular counterpart.

Objective-C is not a proprietary language and is possible to compile with either gcc or Clang/LLVM compilers. However, what is proprietary is its most used standard library, Cocoa.

Cocoa

When Steve Jobs left Apple, he made a smaller company called NeXT. Among other things, they produced a proprietary operating system called NeXTSTEP, based on Unix.[9]

This operating system used Objective-C as its standard language, and proprietary libraries, called OpenStep.[10]

Several years later, Apple (now merged with NeXT) made its new version of Mac OS, called Mac OS X; this operating system was partially based on NeXTSTEP and also used some of its proprietary libraries, now renamed Cocoa.[11]

Cocoa is not the only library for Objective-C, but because Apple is the main investor in Objective-C-based systems, it's a de-facto standard library.

GNUstep

GNUstep is a free re-implementation of OpenStep/Cocoa.[12].

Its development started in the NeXTSTEP days; however, it still hasn't met feature parity with Cocoa's OS X.

Aaron Hillegass in 2nd edition of his popular book *Cocoa Programming on Mac OS X* discouraged people from using GNUStep. He redacted this note in later versions of the book, perhaps because of protests from GNUstep developers[13], but in my opinion, his notes are still valid.

[8] See HILLEGASS; PREBLE, 2011

[9] For a more detailed history, see `https://developer.apple.com/legacy/library/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html#//apple_ref/doc/uid/TP40002974-CH3-SW12`.

[10] Despite the name, OpenStep is not open source – the Open allude to the fact that its API specification was open.

[11] The kernel of Mac OS X is open source, as is its "underlying" operating system called Darwin – however, this system does not contain Cocoa libraries.

[12] See `http://www.gnustep.org/`

[13] `http://www.gnustep.org/resources/BookClarifications.html`

GNUstep implementations are very often buggy and not feature-complete with Cocoa and, most unfortunately, unpredictable. This is what hurt us with Česílko 2.0.

Cocoa and Česílko

When Petr Homola was writing Česílko 2.0, he decided to use Cocoa and Objective-C for development.

On Mac OS X, this configuration is just fine; however, on Linux, where we wanted to run the MT systems (and where only GNUstep is available), this creates unpredictable results.

In my experiments with Czech-to-Slovak translations, I noticed that on Mac OS X, there are about 5-times more sentences generated, than on Linux – while the program was compiled from the same sources.

After thorough inspection, I found out the error was in GNUstep implementation of NSDictionary – Cocoa's implementation of associative array[14] – in some unpredictable cases, NSDictionary returns two different values for two equal NSString keys[15]. As a result, one of the modules returned wrong inflection patterns for a number of words and the morphological analyzer then returned only a fraction of the results.

After a "hacky", but working workaround for this issue, the system returned same correct results on both OS X and Linux. However, I am not at all confident there aren't more similar issues in GNUstep to further develop the system for Russian; fixing the issues of the standard frameworks, copying API of a closed-source library, that's normally very rarely used, is way beyond the scope of this thesis.

Reading the paper BOJAR; GALUŠČÁKOVÁ; TÝNOVSKÝ, 2011, that presents Česílko 2.0 with a very low BLEU, I think the same issue plagued the authors of that paper – it's unprobable the BLEU of the correctly working system would be that low, when in HOMOLA; KUBOŇ; VIČIČ, 2009, the results of Česílko 2.0 were slightly better than of Česílko 1.0.

2.3 Black-box systems

In this chapter, I am describing all the "black-box" systems – that is, without any access to the source code – that we successfully tried.

2.3.1 PC Translator

Description

PC Translator is a commercial translation system from a Czech company LangSoft (`http://www.langsoft.cz/translator.htm`). PC Translator can translate several language pairs, all with Czech on either source or target side.

Authors of PC Translator don't publish any papers or other literature about the system – what can we tell about its functionality is gathered only from its promotional website and from the experiments with the software itself.

[14] `https://developer.apple.com/library/mac/documentation/Cocoa/Reference/Foundation/Classes/NSDictionary_Class/Reference/Reference.html`

[15] it might have to do something with Unicode; however, NSStrings are supposed to be UTF-8 by default

PC translator seems to be purely rule-based. The system seems to work in following steps:

- some (probably rule-based) morphological analysis of the source language
- translation of the lemmas from source language to target language by searching in a large dictionary
- some synthesis of morphological information and the translated lemma

The system doesn't seem to do any kind of reordering. It also doesn't seem to do any analysis on a deeper level, like sentence constituents. Some of the phrases in the dictionary are longer than one word, but not too many of them.

One of the advantages of PC Translator is its large dictionary – however, the dictionary is sometimes choosing very odd and inprobable choices when disambiguating between more possible translations. For example, the English sentence "I like dogs" is translated as "Mám rád kleště", because the term "dog" can be also translated as "kleště"[16]. This can be seen as a proof that PC Translator is a purely rule-based system.

According to its marketing materials, PC Translator v14 uses a Czech-Russian dictionary with above 650.000 words.

Experiments

We found out it's not easy to automate translating with PC Translator. Its GUI is suited for translating by hand, sentence-by-sentence, but not for automated translation of thousands of sentences. Also, by definition, Windows GUI is harder to automate on Linux machine from a script.

However, we were able to work around that, with the help of VMWare Player virtualization software (`http://www.vmware.com/cz/products/player`) and AutoHotkey GUI scripting software, that allows us to emulate screen clicking (`http://www.autohotkey.com/`). Our workflow therefore is:

- on Linux machine, encode the source from UTF-8 to windows-friendly encoding
- encode the source as HTML code
- start a virtual machine with PC Translator pre-installed
- on the start of the virtual machine, run AutoHotkey script from an outer-machine folder (thanks to VMWare shared folders and Windows Startup scripts)
- via this AutoHotkey script, run PC Translator and click on "translate file" feature
- translate the HTML file (also shared in the VMWare shared folder)
- turn off the virtual machine
- turn the file back from HTML and Windows encodings back to UTF-8

[16] from Collins' Dictionary: "dog – 5. a mechanical device for gripping or holding, esp one of the axial slots by which gear wheels or shafts are engaged to transmit torque"

The HTML part is needed because PC Translator had some problems with translating ordinary text files, plus we can pair the translated sentences better thanks to `id` parameters in `div` tags.

We used the newest version of PC Translator available at the time, which is PC Translator v14.

2.3.2 Google Translate

Google Translate is a popular free online translation service by Google, an American web search giant (`http://translate.google.com`). Although Google is producing many academic papers on machine translation, the whole system is still proprietary and we cannot fully inspect it, as in the case of PC Translator.

Google Translate uses mostly statistical approach to machine translation, see for example OCH, 2005[17]. Its results are often seen as "state-of-the-art" in machine translation.

However, thanks to its purely statistical approach, it either needs huge amounts of data for every language pair, or it needs to use so-called "pivot languages"[18] – in the case of Google Translate, it's usually English; specific English word order and English idioms are then re-translated into the target language and sometimes introduce downright wrong translations.

API

To automate Google Translate, we cannot use the website itself, simply because pasting tens of thousands of lines into a browser window usually crashes the browser and is probably against Google Translate's Terms of Use.

There are some workarounds around this, such as "faking" browser environment using some automation tools and/or libraries, but we used more stable option.

Google Translate, apart from being a website, has a paid translation API[19]. We figured out it's not too expensive for our testing purposes, so we ended up paying for the API.[20]

We used a Java library for Google Translate API, called prosaically "google-api-translate-java" (`http://code.google.com/p/google-api-translate-java`).

The tests were done on 3rd May, 2014.[21]

2.3.3 Microsoft Bing Translator

Another online service that we tried is Microsoft Translator/Bing Translator. (In Microsoft's own materials, the system is usually called Bing Translator when referring to the website and Microsoft Translator when referring to the API. I will call it Microsoft Translator further.)

[17] F. J. Och is a head of Google Translate group in Google

[18] See for example KOEHN, 2010

[19] `https://developers.google.com/translate/?hl=cs`

[20] The cost is measured per character on the source side. We used about 3 million characters and paid about 60 dollars. This is rather high for any repeated experiments, but not that high for one-time translation.

[21] I think it's important to note the date of the tests, because the quality of online services might change overtime.

Microsoft Translator is very similar to Google Translate – it is an online website with an easy GUI, and an additional paid API. Again, the team occasionally publishes some scientific papers, but is again otherwise proprietary.

In different experiment, we found out (non-scientifically), that for some language pairs, Microsoft Translator does more post-editation, that seemed a little rule-based (for example, better verb separation in English-to-German translation). However the system as a whole seems similarly statistical as Google Translate.

API

Again, we used Microsoft Translator API (confusingly marketed as a "dataset" inside Windows Azure platform).

The API is slightly more complex than Google's API because of the auto-expiring token, but we used the example PHP script from the API documentation[22].

The pricing is slightly different in Microsoft Translator than in Google Translate, but in general is slightly cheaper. First 2 million letters are for free, next 2 million are for about 40 US dollars.

2.3.4 Yandex Translate

Yandex (`http://www.yandex.ru`) is a Russian search portal that, according to its website[23], generates 61 percent of web search traffic in Russia.

Apart from being a search engine, Yandex offers a variety of other services. One of them is Yandex Translate (`http://translate.yandex.com`)[24] – again, a simple website for automatized translations, similar to aforementioned Google Translate or Microsoft Translator.

I wanted to include Yandex Translate, because – in theory – as a Russian service, it could have better Russian language models and better Russian support in general.

API

Yandex Translate also has a translation API. The API itself is absolutely free and is probably the easiest of the three online services to implement; however, it has strange and vaguely defined usage limits with no way of checking the actual usage.

In our experiments, the API simply stopped returning sentences after approximately 1 million characters per 24 hours. After 24 hour period, the API became usable again.

2.4 Moses

Moses is an open-source machine translation toolkit with GPL licence, developed as a successor to the closed-source Pharaoh system.[25]

The system is very modular and very customizable, which makes it a bit harder to describe. In this section, I will try to describe our Moses set-up; first, I describe the

[22] `http://msdn.microsoft.com/en-us/library/hh454950.aspx`

[23] `http://company.yandex.com/`

[24] Or `http://translate.yandex.ru` for Russian version

[25] See for example KOEHN; HOANG; BIRCH, et al., 2007 or MT, 2013 – but Moses is used so often and so extensively that many other papers describing it can be found

overview of the entire system, and then I further describe some of the elements and our contributions.

2.4.1 Pipeline overview

At the start, we have a bilingual corpora of a given language pair, and bigger monolingual corpora of the target language.

The bilingual corpora has to be prepared by aligning the sentences, so every sentence has exactly one translation. We describe our corpora in the part ???

The sentences are then word-aligned, which means pairing words to their translations. We are using MGIZA++[26]. From this word alignment, Moses learns a so-called *phrase-based translation model*. From the monolingual corpora, we then learn a statistical *language model* – we use SRILM language model[27].

Moses is then used for so-called *decoding* of the information from both the language model and the translation model, which choses the best possible translation, using algorithms like beam-search.

However, for the best translation, we need to tune Moses parameters for optimal results. This is done using so-called *minimum rate error rating* – or MERT for short, which is tuning the parameters on a small separate development set.

After MERT tuning, we finally have working language model, translation model and Moses parameters, which is our complete translation system.

To reiterate, this is our Moses pipeline

- getting sentence-aligned parallel Czech-Russian corpus, plus Russian monolingual corporus
- world-alignment on parallel corpus
- creating phrase-based translation model
- creating Russian language model
- tuning the parameters for Moses decoder

2.4.2 Managing experiments

For managing the steps described further, such as training models, we need some overaching system – steps variously fail, don't compile, don't fit in memory, etc. We would also like to reuse partial results in more experiments.

Moses itself has built-in perl-based experiment managment system, called prosaically Experiment Management System (EMS). However, this system is not very widely used on UFAL.

Instead of EMS, we use instead another perl-based tool called eman (experiment manager). Eman is described well in BOJAR; TAMCHYNA, 2013 or at its website, `http://ufal.mff.cuni.cz/eman`.

Eman breaks down experiment into so-called "steps". Step encapsulates an atomic part of an experiment and can be in one of a few various states. More importantly, step can be dependent on various other states; if a step fails, all steps dependent on it

[26] See GAO; VOGEL, 2008

[27] See STOLCKE; ZHENG; WANG; ABRASH, 2011

automatically fail. The whole experiment is then just another step, dependent on all the necessary substeps.

Step is represented by a directory in a playground directory. Step is created by copying a script, called "seed", from a library of seeds, to a new directory.

In my opinion, while eman itself is well written, I found the seeds themselves hard to read, too repetitive, and with large amount of code copied and pasted over.

For that reason, I tried to rewrite the seeds as perl modules instead of bash scripts for more clarity and reusability. I am, however, not personally sure if my effort in this regard was successful. I decided to use the module `MooseX::Declare`[28], which seemed to us like a modern way to write modules in perl.

Unfortunately, that module is using very difficult-to-understand perl concepts and source code transformations through `Devel::Declare`, and as a result, it takes long to run and, perhaps worse, returns very confusing and undecypherable errors. So as a result of my rewrite, I have seeds with code that's probably easier to read and refractor, but on the other hand, it's slow and produces very opaque errors.

Author of `MooseX::Declare` is now recommending `Moops` module instead for declarative syntax; this module is, however, requiring perl version 14 and above, while on UFAL's network, only perl 10 is installed.

2.4.3 Word alignment

For word alignment, we are using MGIZA++[29], which is a GPL toolkit based on GIZA++[30], which is itself based on models, sometimes called IBM Model 1 to IBM Model 5[31], which is itself based on expectation–maximization algorithm (EM).

IBM Models and the underlying EM algorithms are explained perfectly in Chapter 4 of KOEHN, 2010 or in those slides by the same author – `http://www.inf.ed.ac.uk/teaching/courses/mt/lectures/ibm-model1.pdf`.

GIZA++ is an implementation of those models. MGIZA++ is just its multi-threaded variant, which makes the word alignment slightly faster.

2.4.4 Phrase-extraction

In this step, Moses takes the word alignment from the previous step and learns a so-called "phrase table". Unlike word alignment, phrase extraction spans multiple words on every side in so-called "phrases".

Phrase table consists of list of phrases, their probabilities in both ways of translation, and their lexical weighting – lexical weighting is the probability of the translated phrase counted by individual word pairs. The exact meaning of the numbers is well explained in KOEHN; OCH; MARCU, 2003.

The phrase-table defines a so-called "translation model".

2.4.5 Language model

Language model is a part of the system, that tries to model the probability of a target language sentence alone. It's trained on a monolingual corpus.

[28] `http://search.cpan.org/~ether/MooseX-Declare-0.38/lib/MooseX/Declare.pm`

[29] See GAO; VOGEL, 2008

[30] See OCH; NEY, 2003

[31] See BROWN; PIETRA; PIETRA; MERCER, 1993

We use SRILM, which is an open source language modeling toolkit. (Although it's open-source, it uses its own license, that allows free use only for non-commercial and educational purposes.) Current status of SRILM is described in STOLCKE; ZHENG; WANG; ABRASH, 2011, original design is described in STOLCKE, 2002.

SRILM uses several models, one of them is n-gram word model, described well in KOEHN, 2010[32]. We use n-grams model to the order 3 with words and order 5 with tags (see section 2.4.7). We smooth the models with Kreser-Ney smoothing with Chen and Goodman's modification[33].

2.4.6 Language model interpolation

As described in the section XX, we had more than one monolingual Russian corpora, but we weren't sure of how high quality each of them were and how helpful it would be. For this reason, we used so-call interpolation (also called mixing).

Linear interpolation in general is described for example in GUTKIN, 2000. On a separate heldout data, set of *lambdas* are trained – the resulting probabilities are then just the individual probabilities, multiplied by the lambdas and summed.

Linear interpolation is supported by Moses by undocumented script in the codebase, called `interpolate-lm.perl`, which in turn uses SRILM's undocumented AWK script `compute-best-mix.gawk` and SRILM's `ngram` with `-mix-lm` option[34]. Eman manager then uses these scripts in the `mixlm` seed.

We used a linear interpolation instead of log-linear interpolation simply because we didn't notice the option until later in the project.

2.4.7 Factored translation

Overview

The pipeline, described in the previous sections, translates phrases from the source language to the target language "as is". Only the exact phrases, found on the source side, can be translated to the exact phrases on the target side; and as they are decoded by Moses, only the phrases themselves are taken into account.

However, with morphologically rich languages such as Russian or Czech, this can result in worse translations because of the number of word forms and resulting data sparsity. With so-called factored translation, we can add some morphological information while still keeping the main ideas of phrase-based translation. Factored translation was introduced in KOEHN; HOANG, 2007.

With factored translation, phrased-based approach is extended with morphological (or other) information. We can add additional information (for example, lemma or morphological tag) to either side of the translation, on a word level – this is called a *factor*. Then, instead of training language models and/or translation models on the words alone, we train them on some combination of these factors and then, with the help of Moses that supports factored translation models, combine them together.

[32] chapter 7

[33] See CHEN; GOODMAN, 1996 and `http://www.speech.sri.com/projects/srilm/manpages/ngram-discount.7.html`

[34] See `http://www.speech.sri.com/projects/srilm/manpages/ngram.1.html`

Our factored translation experiments

In a separate set of experiments only on UMC data (described in the section ??), we realized our Moses results have a high OOV rate[35]; this is easily recognizable by Latin script appearing in Czech-to-Russian translation (or Cyrillics in the opposite direction). We then tried to compare several set-ups for factored tranlation to get lower OOV rate and higher BLEU scores.

We used a set-up described for example in KOEHN; HADDOW, 2012. However, since we did not have Russian morhpology fully working, we used only the system described as `lemma backoff` – with the exception of not translating to lemma. We were not using interpolated backoff, simply because regular backoff is easier to use with Moses.

The main model translates from a word form on the source side to word form and tag on the target side. The backoff model translates from a lemma (or a stem – see below) to form and tag on the target side.

For Russian, we used TreeTagger software[36] with a Russian parameter file[37]. TreeTagger is a closed-source software with a restrictive license, but for free for research purposes.

For Czech, we used tokenizer from UFAL project Treex (described further in section XXX) and morphological analyzer Morče (`http://ufal.mff.cuni.cz/morce/references.php`); however, we didn't actually use its output, as described further.

When we experimented further, we found out that using not lemma on the source side, but a very crude lexical stem – just using the first 4 letters of a word – gets better results[38]. When we tried the experiment with different lengths of the stem, we got the results, shown in the graph XX – the lexical stem of the lengths XX was the best.

Therefore, we continued to use that in our further experiments, as for example in the WMT submissions[39].

[35] Out Of Vocabulary; how many words were untranslated due to not being found in the phrase table

[36] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`, also see SCHMID, 1994 and SCHMID, 1995

[37] trained on a corpus created by Serge Sharoff, see `http://corpus.leeds.ac.uk/mocky/`

[38] Using stems instead of lemmas is suggested for example in XX. However, their stems are more linguistically motivated, while we just crudely take first few letter

[39] citace tu

Conclusion

Bibliography

ANDERSEN, Henning. 2013. On the Origin of the Slavic Aspects: Aorist and Imperfect. *Journal of Slavic Linguistics*. 2013, vol. 21, no. 1, pp. 17–43. Available also from WWW: ⟨http://www.questia.com/library/journal/1G1-332655163/on-the-origin-of-the-slavic-aspects-aorist-and-imperfect⟩.

ARCHIBALD, Eric. 2008. *New Testament Greek Course* [online]. Swindon : Hayes Press, 2008 [visited on 2014-04-18]. Available from WWW: ⟨http://www.hayespress.org/biblehelps-GREEK-30⟩.

BÍLEK, Karel; KLYUEVA, Natalia; KUBOŇ, Vladislav. 2013. Exploiting Maching Learning for Automatic Semantic Feature Assignment. In BOONTHUM-DENECKE, Chutima; YOUNGBLOOD, G. Michael (ed.). *FLAIRS Conference*. 2013. Available also from WWW: ⟨http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS13/paper/view/5922⟩.

BIBLICA. 1973, 1978, 1984, 2011. *The Holy Bible, New International Version, NIV* [online]. Colorado Springs : Biblica, 1973, 1978, 1984, 2011 [visited on 2014-04-18]. Available from WWW: ⟨http://www.biblegateway.com/passage/?search=1+Corinthians+1⟩.

BOJAR, Ondřej; HOMOLA, Petr; KUBOŇ, Vladislav. 2005. An MT System Recycled. In. *Proceedings of MT Summit X*. 2005, pp. 380–387. Available also from WWW: ⟨http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.103.8336\&rep=rep1\&type=pdf⟩. ISBN 974-7431-26-2.

BOJAR, Ondřej. 2012. *Čeština a strojový překlad (Czech Language and Machine Translation)*. Praha, Czech Republic : ÚFAL, 2012. Studies in Computational and Theoretical Linguistics. ISBN 978-80-904571-4-0.

BOJAR, Ondřej; GALUŠČÁKOVÁ, Petra; TÝNOVSKÝ, Miroslav. 2011. Evaluating Quality of Machine Translation from Czech to Slovak. In LOPATKOVÁ, Markéta (ed.). *Information Technologies – Applications and Theory*. 2011, pp. 3–9. ISBN 978-80-89557-01-1.

BOJAR, Ondřej; TAMCHYNA, Aleš. 2013. The Design of Eman, an Experiment Manager. *The Prague Bulletin of Mathematical Linguistics*. 2013, vol. 99, pp. 39–58. ISSN 0032-6585.

BROWN, Peter F.; PIETRA, Vincent J. Della; PIETRA, Stephen A. Della; MERCER, Robert L. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.* 1993, vol. 19, no. 2, pp. 263–311. Available also from WWW: ⟨http://acl.ldc.upenn.edu/J/J93/J93-2003.pdf⟩. ISSN 0891-2017.

CHEN, Stanley F; GOODMAN, Joshua. 1996. An empirical study of smoothing techniques for language modeling. In. *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Santa Cruz, California : Association for Computational Linguistics, 1996, pp. 310–318. ACL '96. Available also from WWW: ⟨http://www.speech.sri.com/projects/srilm/manpages/pdfs/chen-goodman-tr-10-98.pdf⟩.

COLMERAUER, Alain. 1970. *Les systèmes-Q; ou un formalisme pour analyser et synthétiser des phrases sur ordinateur*. 1970.

FLEK, Alexandr et al. 2012. *Bible, překlad 21. století* [online]. 2012 [visited on 2014-04-18]. Available from WWW: ⟨http://onlineb21.bible21.cz/bible.php?kniha=1korintskym⟩.

GAO, Qin; VOGEL, Stephan. 2008. Parallel Implementations of Word Alignment Tool. In. *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*. Columbus, Ohio : Association for Computational Linguistics, 2008, pp. 49–57. SETQA-NLP '08. Available also from WWW: ⟨http://www.aclweb.org/anthology/W08-0509.pdf⟩. ISBN 978-1-932432-10-7.

GUTKIN, Alexander. 2000. *Log-linear interpolation of language models*. 2000. Available also from WWW: ⟨http://www.cstr.ed.ac.uk/downloads/publications/2000/gutkin_mphil.pdf⟩.

HAJIČ, Jan. 1987. RUSLAN: An MT System Between Closely Related Languages. In. *Proceedings of the Third Conference on European Chapter of the Association for Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1987, pp. 113–117. EACL '87. Available also from WWW: ⟨http://dx.doi.org/10.3115/976858.976879⟩.

HAJIČ, Jan; HRIC, Jan; KUBOŇ, Vladislav. 2000. Machine Translation of Very Close Languages. In. *Proceedings of the Sixth Conference on Applied Natural Language Processing*. Seattle, Washington : Association for Computational Linguistics, 2000, pp. 7–12. ANLC '00. Available also from WWW: ⟨http://www.aclweb.org/anthology/A00-1002⟩.

HILLEGASS, Aaron; PREBLE, Adam. 2011. *Cocoa Programming for Mac OS X*. Fourth. 2011. Available also from WWW: ⟨https://www.bignerdranch.com/book/cocoa_programming_for_mac_os_x_th_edition_⟩. ISBN 9780132902205.

HOMOLA, Petr; KUBOŇ, Vladislav; VIČIČ, Jernej. 2009. Shallow Transfer Between Slavic Languages. In. *Proceedings of Balto-Slavonic Natural Language Processing*. Kraków, Poland : Polska Akademia Nauk, 2009, pp. 219–232. ISBN 978-83-60434-59-8.

KOEHN, Philipp. 2010. *Statistical Machine Translation*. Cambridge : Cambridge University Press, 2010. Statistical Machine Translation. Available also from WWW: ⟨http://www.statmt.org/book/⟩. ISBN 9780521874151.

KOEHN, Philipp; HADDOW, Barry. 2012. Interpolated backoff for factored translation models. In. *Proceedings of the Tenth Conference of the Association for Machine Translation in the Americas (AMTA)*. 2012.

KOEHN, Philipp; HOANG, Hieu. 2007. Factored Translation Models. In. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic : Association for Computational Linguistics, 2007, pp. 868–876. Available also from WWW: ⟨http://homepages.inf.ed.ac.uk/pkoehn/publications/emnlp2007-factored.pdf⟩.

KOEHN, Philipp; HOANG, Hieu; BIRCH, Alexandra, et al. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In. *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. Prague, Czech Republic : Association for Computational Linguistics, 2007, pp. 177–180. ACL '07. Available also from WWW: ⟨http://acl.ldc.upenn.edu/P/P07/P07-2045.pdf⟩.

KOEHN, Philipp; OCH, Franz Josef; MARCU, Daniel. 2003. Statistical Phrase-based Translation. In. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*. Edmonton, Canada : Association for Computational Linguistics, 2003, pp. 48–54. NAACL '03. Available also from WWW: ⟨http://acl.ldc.upenn.edu/N/N03/N03-1017.pdf⟩.

MT, Moses. 2013. *Moses/Overview* [online]. 2013 [visited on 2014-05-10]. Available from WWW: ⟨http://www.statmt.org/moses/?n=Moses.Overview⟩.

NGUYEN, Hong-Thai. 2009. *Des systèmes de TA homogènes aux systèmes de TAO hétérogènes*. 2009. Available also from WWW: ⟨http://hal.inria.fr/docs/00/44/75/71/PDF/these_Hong-Thai_NGUYEN_UJF_TA-et-TAO-heterogenes.pdf⟩.

OCH, Franz Josef. 2005. *MT Summit X*. Phuket : Google, 2005.

OCH, Franz Josef; NEY, Hermann. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Comput. Linguist.* 2003, vol. 29, no. 1, pp. 19–51. Available also from WWW: ⟨http://acl.ldc.upenn.edu/J/J03/J03-1002.pdf?q=modles⟩. ISSN 0891-2017.

OFFORD, Derek. 1996. *Using Russian: A Guide to Contemporary Usage*. Cambridge : Cambridge University Press, 1996. Available also from WWW: ⟨http://books.google.cz/books?id=iWy0clZRkQwC⟩. ISBN 9780521457606.

OLIVA, Karel. 1989. *A Parser for Czech Implemented in Systems Q*. Prague : Matematicko-fyzikální fakulta UK, 1989. Explizite Beschreibung der Sprache und automatische Textbearbeitung.

PARKER, Micheal W. 2009. *Hellenistic Greek* [online]. 2009 [visited on 2014-04-18]. Available from WWW: ⟨http://greek-language.com/grammar/22.html⟩.

RINGE, Don. 2008. *From Proto-Indo-European to Proto-Germanic*. Oxford : Oxford University Press, 2008. A Linguistic History of English. Available also from WWW: ⟨http://ukcatalogue.oup.com/product/9780199552290.do⟩. ISBN 9780199552290.

SCHMID, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In. *Proceedings of international conference on new methods in language processing*. 1994, pp. 44–49. Available also from WWW: ⟨http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger1.pdf⟩.

SCHMID, Helmut. 1995. Improvements In Part-of-Speech Tagging With an Application To German. In. *In Proceedings of the ACL SIGDAT-Workshop*. 1995, pp. 47–50. Available also from WWW: ⟨http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/data/tree-tagger2.pdf⟩.

SIEWIERSKA, Anna; UHLÍŘOVÁ, Ludmila. 1998. An overview of word order in Slavic languages. *Constituent Order in the Languages of Europe*. 1998, vol. 20, no. 1, pp. 105. Available also from WWW: ⟨http://www.degruyter.com/viewbooktoc/product/4280⟩.

SPOUSTOVÁ, Drahomíra; SPOUSTA, Miroslav; PECINA, Pavel. 2010. Building a Web Corpus of Czech. In. *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC 2010)*. Valletta, Malta : European Language Resources Association, 2010, pp. 998–1001. Available also from WWW: ⟨http://www.lrec-conf.org/proceedings/lrec2010/pdf/810_Paper.pdf⟩. ISBN 2-9517408-6-7.

STOLCKE, Andreas. 2002. SRILM-an extensible language modeling toolkit. In. *Proceedings International Conference on Spoken Language Processing*. 2002, pp. 257–286. Available also from WWW: ⟨http://www.speech.sri.com/cgi-bin/run-distill?papers/icslp2002-srilm.ps.gz⟩.

STOLCKE, Andreas; ZHENG, Jing; WANG, Wen; ABRASH, Victor. 2011. SRILM at sixteen: Update and outlook. In. *Proceedings IEEE Automatic Speech Recognition and Understanding Workshop*. 2011. Available also from WWW: ⟨http://t3-1.yum2.net/index/www.speech.sri.com/papers/asru2011-srilm.pdf⟩.

SUSSEX, Roland; CUBBERLEY, Paul. 2011. *The Slavic Languages*. Cambridge : Cambridge University Press, 2011. Cambridge Language Surveys. Available also from WWW: ⟨http://www.cambridge.org/us/academic/subjects/languages-linguistics/european-language-and-linguistics/slavic-languages⟩. ISBN 9780521294485.

WIKISOURCE. 2013. *Biblia Sacra Vulgata (Stuttgartensia)/ad Corinthios I — Wikisource,* [online]. 2013 [visited on 2014-04-18]. Available from WWW: ⟨http://la.wikisource.org/w/index.php?title=Biblia_Sacra_Vulgata_(Stuttgartensia)/ad_Corinthios_I\&oldid=62334⟩.

List of Tables

List of Abbreviations

Attachments