REYKJAVÍK UNIVERSITY
HÁSKÓLINN Í REYKJAVÍK

Improving the tagging accuracy of Icelandic text

Ida Kramarczyk

Master of Science
June 2009

Reykjavík University - School of Computer Science

M.Sc. Thesis

Improving the tagging accuracy of Icelandic text

by

Ida Kramarczyk

Thesis submitted to the School of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Master of Science

June 2009

Thesis Committee:

Dr. Hrafn Loftsson, supervisor
Assistant Professor, Reykjavík University

Eiríkur Rögnvaldsson
Professor, University of Iceland

Dr. Hannes Högni Vilhjálmsson
Assistant Professor, Reykjavík University

The undersigned hereby certify that they recommend to the School of Computer Science at Reykjavík University for acceptance this thesis entitled **Improving the tagging accuracy of Icelandic text** submitted by Ida Kramarczyk in partial fulfillment of the requirements for the degree of **Master of Science**.

_____

Date

_____

Dr. Hrafn Loftsson, supervisor
Assistant Professor, Reykjavík University

_____

Eiríkur Rögnvaldsson
Professor, University of Iceland

_____

Dr. Hannes Högni Vilhjálmsson
Assistant Professor, Reykjavík University

The undersigned hereby grants permission to the Reykjavík University Library to reproduce single copies of this thesis entitled **Improving the tagging accuracy of Icelandic text** and to lend or sell such copies for private, scholarly or scientific research purposes only.

_____

Date

_____

Ida Kramarczyk
Master of Science

Improving the tagging accuracy of Icelandic text

by

Ida Kramarczyk

June 2009

Abstract

In this thesis, four attempts to improve the tagging accuracy for Icelandic text are presented. All of them were tested on IceTagger, a linguistic rule-based tagger with a tagging accuracy of 91.59%, and TnT, a data-driven tagger with a tagging accuracy of 90.45% for Icelandic. The first attempt was to reduce the number of tags in the Icelandic tagset. Various different reductions were tested. The set which gave the best result improved the tagging accuracy for IceTagger by 1.19% and for TnT by 1.45%. The second attempt was to use a larger dictionary which improved tagging by 0.56% for IceTagger and 0.69% for TnT. The third attempt was to improve tagging accuracy by integrating a lemmatizer for Icelandic into IceTagger to use for unknown wordforms of words which already appear in the lexicon in a different form. This did not show any noteworthy results. The last attempt was a combination of taggers. We used 7 taggers, IceTagger, BI+WC+CT, TnT, fnTBL, TreeTagger, MBT and MXPOST, and tested various combinations of them. The best combination, consisting of 5 taggers, gave a tagging accuracy of 93.74%, and 94.14% using a bigger dictionary. Lastly, the best combination, using a bigger dictionary and a reduced tagset, resulted in 94.99% accuracy.

Aukin mörkunarnákvæmni íslensks texta

eftir

Ida Kramarczyk

Júní 2009

Útdráttur

Fjórar aðferðir voru notaðar í þessu verkefni til að hækka nákvæmni markara fyrir íslenskan texta. Allar fjórar aðferðinar voru prófaðar á IceTagger, sem er málfræðilegur reglumarkari, en hann nær 91.59% nákvæmni og svo TnT, sem er gagnamarkari sem nær 90.45% nákvæmni fyrir íslensku. Fyrsta aðferðin var að minnka stærð íslenska markamengisins. Nokkrir möguleikar á minnkuðu markamengi voru prófaðir en breytingar á markamenginu sem ákveðnar voru hækkuðu nákvæmni um 1.19% fyrir IceTagger og um 1.45% fyrir TnT. Önnur aðferðin var að nota stærra orðasafn sem hækkaði nákvæmni um 0.56% fyrir IceTagger og um 0.69% fyrir TnT. Þriðja aðferðin var að setja lemmara inn í IceTagger til að leita að lemmu óþekktra orðmynda og fletta því svo upp í orðasafninu. Þetta bar engan árangur. Fjórða aðferðin var að sameina sjö mismunandi markara: IceTagger, BI+WC+CT, TnT, fnTBL, TreeTagger, MBT og MXPOST. Við prófuðum marga möguleika og fundum að besti árangur fékkst með samsetningu 5 markara. Nákvæmni hækkaði í 93.74% en 94.14% með notkun á stærra orðasafninu. Að lokum, með því að nota besta sameinaða markarann, stærri orðabók og minkað markamengi jókst nákvæmni í 94.99%.

Acknowledgements

This work was supported by Rannís with the research grant nr. 070025023 "Aukin mörkunarnákvæmni íslensks texta" ('Improving the tagging accuracy of Icelandic text').
I want to thank my mum for suggesting a trip to Iceland in 2001 which changed my life. Without this trip and her constant support over the last 5 years I would not be where I am today.
I want to thank my supervisor, Hrafn Loftsson for his advice and Hannes Högni Vilhjálmsson for the motivating words I badly needed.
Furthermore I want to thank Hilmar Haukur Guðmundsson, Renate Polak and Deena Bollinger-Kramarczyk for their corrections on my thesis, and Sigrún María Ammendrup for correcting my Icelandic. I want to thank all my friends and colleagues for their help, support and understanding over the last two very stressful years. You made those years an unforgettable experience.
And finally, very special thanks go to my friend Cornelia Panzenböck, who not only helped me throughout the whole study but is also the only person besides the members of the committee who read my thesis more than once and made many helpful suggestions and corrections to it. I could not have done this without you.

Publications

Some of the material presented in this thesis has appeared in the following publication: Hrafn Loftsson, Ida Kramarczyk, Sigrún Helgadóttir and Eiríkur Rögnvaldsson. Improving the PoS tagging accuracy of Icelandic text. *Proceedings of the* 17^{th} *Nordic Conference of Computational Linguistics (NoDaLiDa-2009)* (pp. 103-110). Odense, Denmark.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Language Technology in Iceland is relatively new. Its beginning was marked in 1998 with a report written by a committee, appointed by the Minister of Education, Science and Culture (Ólafsson, Rögnvaldsson, & Sigurðsson, 1999)[1]. The purpose of the report was to identify the main components which are essential for using Icelandic in IT.
In 2008, Rögnvaldsson (2008) published the results of what has been achieved since the original report. In this section, I will summarize what has been done up to now in Language Technology in Iceland and what is still left to do.

1.1 The Goals of 1999

As mentioned above, the committee suggested some points, 8 in number, to make Icelandic usable in everyday life. Those points are:

- The main computer programs and operating systems should be available in Icelandic.
- Icelandic letters should be available in all media (e.g. computer, mobile phone).
- Work on the parsing of Icelandic text should be continued.
- Good utility programs should be developed for Icelandic.
- A good Icelandic speech synthesizer should be developed.

[1] See http://www.tungutaekni.is/news/Skyrsla_en.pdf for an English translation of the conclusions.

- A speech recognition tool should be developed which understands normal Icelandic speech.
- Translation programs between Icelandic and other languages should be developed.
- Institutions and companies will be entrusted with special projects.

Work has started in all of these fields (see also chapter 3.1), although progress in some was greater than in others. Some applications rely very much on basic tools such as taggers and parsers, which have been developed but have not met the high requirements of their future applications yet. In order to be able to parse Icelandic text, translate Icelandic text or understand spoken Icelandic we need to get reliable grammatical information about the text we are working with.

1.2 Why tagging?

A tagger is a tool that takes continuous text in a natural language as an input, analyses it and returns the text including grammatical analysis for every single word. Grammatical information is added as tags which follow the word. They contain information about the word class (noun, verb, adjective, ...) and depending on the class it can contain further information. As an example, the Icelandic word *hestur* ('horse') would have the tag *nken* which stands for noun (n), masculine (k), singular (e), nominative (n). The Icelandic tagset will be described in more detail in section 3.2.

Without this information a parser would not be able to analyze the syntactic structure of a sentence. Grammar and syntax analysis provide the necessary information to understanding the construction of a sentence. They are the basics for many applications such as grammar and spell checking, speech systems (see chapter 3.1.3) or information extraction.

A tagger is one of the basic parts of a BLARK, a Basic Language Resource Kit[2]. The idea of a BLARK was first presented in (Krauwer, May 1998) where the minimal requirements were named as:

[2] Described by Krauwer (2003) as "the minimal set of language resources that is necessary to do any precompetitive research and education at all".

- A minimal general text corpus to be able to do any precompetitive research for the language at all, say (as an arbitrary example) 10 million words of recent newspaper text, annotated according to some generally accepted standards
- Something similar for a spoken text corpus
- A collection of basic tools to manipulate and analyze the corpora
- A collection of skills that constitute the minimal starting point for the development of a competitive NL/Speech technology industry

These requirements have been extended since and a new, longer description can be found in (Krauwer, 2003). These requirements are never complete. They can vary between languages because every language can have their own special requirements that other languages do not have to meet.

The current Icelandic BLARK is fairly small. It contains the IceNLP which is a toolkit consisting of a tagger and a parser especially developed for Icelandic (Loftsson & Rögnvaldsson, 2007a). This toolkit is only a few years old. In (Ólafsson et al., 1999) the development of such tools was a future plan and even in (Rögnvaldsson, 2008) it is only mentioned briefly as a first step towards a BLARK.

What is mostly missing are dictionaries and corpora, but those are often privately owned and getting a license for each of them can be costly and difficult. The only available PoS tagged corpus for Icelandic is Íslensk Orðtíðnibók (Pind, Magnússon, & Briem, 1991) (the Icelandic Frequency Dictionary, (IFD)). A large lexical database is available for Icelandic, Beygingarlýsing íslensks nútímamáls ((BÍN), the Inflections of modern Icelandic) (Bjarnadóttir, 2005), which contains a list of inflections of Icelandic words. The dictionaries will be discussed in more detail in section 3.3.

1.3 Increasing the Tagging Accuracy

The motivation for this project originated from the relatively low tagging accuracy reached for Icelandic so far. IceTagger, a rule-based tagger, reached an accuracy of 91.54% (Loftsson, 2008b), trained on the IFD corpus. As of today, IceTagger is the only tagger especially developed for tagging Icelandic text. Data-driven taggers, that can be trained on Icelandic, have also been tested. The best one, TnT, a statistical tagger, reached an accuracy of only 90.44% (Loftsson, 2006b; Helgadóttir, 2005), also trained on the IFD. Through combinations of several taggers an accuracy of 93.48% was reached (Loftsson, 2006b). These numbers sound high but compared to other

languages they are rather low. For Swedish an accuracy of 93.55% (Megyesi, 2002) has been reached and for English and German an even higher accuracy of 96-97% (Brants, 2000) has been reached.

The goal of this project is to **develop methods to increase the accuracy for tagging Icelandic text**. My supervisor Hrafn Loftsson submitted this project to Rannís (the Icelandic Center for Research) who issued a master's research grant. The project, as offered to me consists of four parts:

1.3.1 Part 1 - New Tagset

In the literature, a correlation between the size of the tagset and the accuracies reached by taggers has been pointed out (Loftsson, 2008b). Therefore, the first approach to increase the tagging accuracy was to scale down the Icelandic tagset. The main tagset for Icelandic consists of 700 tags as opposed to 139 tags in Swedish (Megyesi, 2002) or 45 tags used in the English tagset of the Penn Treebank (Marcus, Santorini, & Marcinkiewicz, 1993).

We used two approaches to test our hypothesis: internal and external mapping. Internal mapping changes the actual tagset the tagger is using. The tagset that the tagger is trained on is reduced so text is tagged with less detailed information. In contrast, external mapping does not change the tagset itself. The tagger is still trained on the larger tagset and applies the big tagset during tagging. The reduction is applied after tagging is complete where the larger tagset is mapped to a smaller one, losing depth of information. The advantage of external mapping is that the information at runtime is not lost. The tagger can thus use the full amount of information for disambiguation. However, before the tagger returns the tagged text, the tags are mapped to a smaller tagset. Errors in tagging details that are not relevant to an application can be hidden in that way. Furthermore, different external tagsets can be applied depending on the needs of an application but the training corpus does not need to be adjusted whereas internal mapping would require an adapted corpus for every reduced tagset.

The smaller tag set with only 450 tags, that resulted from this experiment, improved the tagging accuracy for IceTagger by 1.18% and of TnT by 1.44%.

1.3.2 Part 2 - Bigger Dictionary

The tagging accuracy for unknown words is much lower than it is for known words. IceTagger, for example reaches 92.74% accuracy for known words but only 75.09% for unknown words (Loftsson, 2008b). In order to reduce the number of unknown words, we added the Inflections of Modern Icelandic (BÍN) to the dictionary we derived from the IFD. This reduced the unknown word ratio from 6.79% to 1.15% on average.

As a result, the tagging accuracy for known and unknown words decreased. A reason for that would be the shift in previously unknown words to the known words (5.64%) due to the larger dictionary. The 1.15% of unknown words that were left were words with tags that were difficult to guess. On the other hand, the newly added words in the dictionary had no frequency information, so choosing the right tag in a tag profile was made difficult for IceTagger which reduced the tagging accuracy for known words as well. Nevertheless, the overall tagging accuracy increased by 0.54% for IceTagger and 0.69% for TnT.

1.3.3 Part 3 - Improving the Tagging Accuracy

Many Icelandic verbs govern the case of their objects. IceTagger uses an automatically extracted list from the IFD of verb and object case pairs to determine the correct case. The list only contains the word forms of the verbs that have been found during training. We used a lemmatizer to change those word forms into the infinitive form of the verb. In this way, verbs can be found in the list even if the specific word form has not occurred during training. Furthermore, we added a second list of verb-object case pairs also extracted from the IFD, but by a script, designed to pick out words from sentences of a certain structure (for more details, see chapter 4.3). We then integrated the lemmatizer into IceTagger to change verbs into their infinitive forms before look up. This approach did not show significant improvement.

1.3.4 Part 4 - Tagger Combinations

Different taggers make different kinds of errors. Combining a number of taggers can greatly improve tagging accuracy. We used a combination of simple and weighted voting (see chapter 2.4) to combine between five and seven taggers. In simple voting every tagger has one vote and each vote has the same weight. In weighted voting

some taggers have a stronger vote than others. In our case, we used simple voting but if the result was undecided (tie) the best three taggers got stronger votes. If the result was still undecided, the tag which IceTagger gave was chosen. These experiments resulted in a tagging accuracy of 93.74% for a combination of five taggers and 94.15% when adding parts of the lexical database BÍN to the dictionary.

1.4 Outline

This thesis is organized as follows. An overview of the different types of taggers and all the taggers used in this project will be given in chapter 2. Chapter 3 gives a brief review of the work in the field of language technology in Iceland so far. I will briefly discuss the Icelandic tagset in 3.2, the corpora available for Icelandic in 3.3, and the method used for evaluation in this project in 3.4.1. Chapter 3 ends with the results from previous tagging approaches for Icelandic. In chapter 4, I will present in detail the results of the 4 parts of this project: the use of a smaller tagset (4.2), the use of a larger dictionary (4.1), an improved version of IceTagger (4.3) and the combination of taggers (4.4). Chapter 5 will give a short summary about the findings of the project.

Chapter 2

PoS-Tagging

There are two main types of taggers: linguistic rule-based taggers and data-driven taggers. The first type uses linguistic rules, specially developed for a single language, whereas the latter can be trained on an annotated corpus of virtually any language.

A tagger assigns one or more tags to a word. The set of possible tags for a word is called a tag profile. In order to choose one tag out of the whole profile the tagger needs to carry out disambiguation, which means making a decision for one tag over an ambiguous set of possible tags. This can be done with either rules or probabilities calculated from the training corpus.

Data-driven taggers rely on a big annotated corpus for training, but can be trained on any language. Rule-based taggers contain language specific rules. The development is said to be time consuming (Hagen, Johannessen, & Nøklestad, 2000) and a tagger of this type can only be used for one specific language. On the other hand, data-driven taggers have a limited window size[1] whereas a rule can function on a whole sentence or even beyond the scope of a sentence.

We used seven taggers for our experiment. Two of them, IceTagger, a linguistic rule-based tagger, and TnT, a statistical part-of-speech tagger, were used throughout the whole project and will be described in more detail. The other 5 taggers were used only for testing different combinations of taggers and will therefore only be described briefly here.

[1] A window is a part of the sentence which is looked at. ±2 words relative to the focus word would for example result in a 5 word window.

2.1 Rule-Based Taggers

2.1.1 Constraint Grammar

The formalism of Constraint Grammar is probably the best known rule-based tagging method. It has been developed by Karlsson, Voutilainen, Heikkilä, and Anttila (1995) and has been used since to develop taggers for various languages, e.g. French (Chanod & Tapanainen, 1994) and Norwegian (Hagen et al., 2000).

The basic idea of a constraint grammar is that it should be able to analyze unrestricted text and return the best possible disambiguation, preferably fully disambiguated text. One of the foundations of a constraint grammar is that not only morphological disambiguation is carried out by constraints but also syntactic disambiguation. Constraints are rules that are applied to a word and depending on their scope they can take more or less of the surrounding information into account. Karlsson et al. (1995) distinguishes between two types of grammatical ambiguities to which constraints are applied:

- Local ambiguity: is solvable by looking at just a few adjacent words, also called a window
- Global ambiguity: concern the whole sentence

Ambiguous tags are then deleted if they match the constraints. An example of a local constraint would be *The preceding word is the "sign of infinitive" and the focus word has tags which are not of a verb* then these tags that fit the constraint, which are all non-verb tags, are deleted. It is also possible to phrase constraints in a positive way so that the matching tags are kept and all other tags are deleted. A global constraint is used in a similar way, but can go much further. It is not restricted to a fixed window size like a local constraint.

Most importantly, when applying those rules is that at least one tag has to be left after disambiguation. The constraints should not delete all possible tags but eliminate tags until (in the best case) only one disambiguated tag is left.

2.1.2 IceTagger

IceTagger is a linguistic rule-based tagger which was developed by Loftsson (2008b). It uses the idea of constraints from Constraint Grammar. It contains 175 local rules (local constraints) and a set of heuristics (global constraints) to force feature

agreement. Developing a rule-based tagger is usually a time consuming task. Due to the combination of heuristics and just a few local rules the development time was only 7 man months. A tagging accuracy of 91.59% was reached. More thorough descriptions can be found in (Loftsson, 2007b, 2008a, 2008b).

IceTagger derives its dictionary from the IFD corpus. In later experiments we combined this dictionary with entries from the lexical database, BÍN. This will be discussed in more detail in chapter 4.1.

IceTagger takes a text file with one word per line as an input which has to be prepared beforehand. This happens in the preprocessor where tokenisation is carried out. Here the text is broken down into its basic tokens (words, numbers and punctuation marks). Sometimes sentence segmentation has to be carried out additionally in case it's unclear where one sentence ends and another one starts.

The preprocessor is followed by the 3 main components of IceTagger: a morphological analyzer, local rules and heuristics. These parts will be explained in more detail below.

2.1.2.0.1 IceMorphy

IceMorphy is the morphological analyzer for IceTagger. When getting a tokenized text as input, IceMorphy looks up each word in the dictionary and returns the tag profile, if one exists.
If a word is not found in the dictionary, IceMorphy, which also functions as a unknown word guesser, is guessing a tag based on morphological, compound and ending analysis. First, the morphological analyzer tries to identify the morphological class of the unknown word using its morphological ending. A word w has the form $w = stem + ending$. The stem of a word is the part of the word that does not change during declension[2]. An example is *hestur* 'horse'. It declines in singular as follows:

hestur	-	*hest*	-	*hesti*	-	*hests*
(nom.)		*(acc.)*		*(dat.)*		*(gen.)*

The stem here would be *hest* because this part of the word always stays the same. It is not possible to remove either a prefix or a suffix from it.

Once the stem has been separated from the word, IceMorphy generates all possible endings for this stem assuming a morphological class. Inside this class, the unknown

[2] This is not always correct because vowels might change depending on the ending that is added, but as a general idea we can use this as an explanation.

word has its tag. As an example, assume *hests* is an unknown word. By finding the stem and assuming the morphological class of *hestur*, the masculine genitive singular tag for *hests* is already preassigned.

All resulting words from this morphological class are then looked up in the dictionary until one is found. If none of the words appears in the dictionary and more than one morphological class could have been assumed, the next one is chosen and again all possible endings are generated and the words looked up in the dictionary. This is continued until either a word has been found in the dictionary or all possible words have been looked up without success.

If the morphological analysis was not successful, IceMorphy continues with a compound analysis. Here prefixes are removed from the word and the new word is sent again to morphological analysis. If this search returns a tag profile, the original word, including the prefix, is assigned the same tag profile.

In case neither analyzer could find a tag profile matching the unknown word the ending analyzer is going to look up the ending in an endings dictionary which contains the most common endings for nouns, adjectives and verbs.

IceMorphy also includes a tag profile gap filling component. Due to the relatively small size of the IFD corpus a lot of tags are missing in the tag profiles. This component can fill the gaps by analyzing the existing profile of the word. As an example we can look at the word lampi (the lamp):

lampi	-	*lampa*	-	*lampa*	-	*lampa*
(nom.)		*(acc.)*		*(dat.)*		*(gen.)*

For every weak noun in Icelandic, the genitive, dative and accusative look the same. If IceMorphy finds a word of this form with missing tags for one or more of these cases it's going to fill these gaps with the appropriate tags.

2.1.2.0.2 Local Rules

Local rules take into consideration the context of a word, e.g. they define the surrounding in which the word is supposed to be found. These rules then eliminate ambiguous tags one by one. However, a local rule can only eliminate tags from the focus word itself and not from any other word in the context it is looking at.

Two examples for local rules from (Loftsson, 2008b) are:

$$L_1.isOnlyWordClass(x) \text{ AND } L_2.isOnlyWordClass(y)$$

$$R_1.isWordClass(x) \text{ OR } R_2.isWordClass(y)$$

L_1/L_2 denote a token 1 and 2 respectively to the left of the focus word and R_1/R_2 denote the same to the right. isOnlyWordClass(x) is checking if the possible tags of the word are only from word class x whereas isWordClass(x) is checking if a tag of word class x is one of the possible tags. If a condition is met, the tag that applies to it is eliminated. A description of the local rules and more examples can be found in (Loftsson, 2008b).

Before local rules are applied, idioms and phrasal verbs are tagged unambiguously. A list of them is kept in special dictionaries. Local rules are then used for disambiguation. If a word has more than one tag, IceTagger tries to apply local rules to the tags of this word. The tags are picked one by one, starting with the first one (most frequent). The rules are sorted by word class. Depending on the class of the tag (verb, noun, ...) a different set of rules is chosen. IceTagger then works its way through this set of rules.

2.1.2.0.3 Heuristics After the local rules have been applied, each sentence is separated into clauses at commas, semicolons and conjunctions. Now the heuristics continue the disambiguation process. They tag grammatical functions, prepositional phrases and force feature agreement[3]. Heuristics perform the following steps:

1. mark prepositional phrases (PP)
2. mark verbs
3. mark subjects
4. force subject-verb agreement
5. mark object
6. force subject-object agreement
7. force verb-object agreement
8. force nominal agreement

[3] To force feature agreement means to make sure that i.e. an adjective and a noun agree in gender, number and case or a subject and a verb agree in person and number.

9. force PP agreement

The first heuristic checks to see if a prepositional tag has a higher frequency in the IFD than the other tags. The tag with the highest frequency is always the first in line because the tags are sorted by descending frequency in the dictionary derived from the IFD. If a prepositional tag is found in this place, the word is marked with a syntactical tag PP and all non-prepositional tags are deleted. Nominals following this word are also marked PP in case of feature agreement with the preposition.

The second heuristic marks verbs. Again it searches for words which have a verb tag as the most frequent tag. If a word like that is found, all non-verb tags are deleted.

The next heuristic is marking subjects of verbs. Icelandic has a free word order. A subject can basically stand anywhere in a sentence while a verb has to be in second position, but can be moved around more freely for rhetorical reasons. However, the most common word order is SVO[4], so first it looks for a subject to the left of the verb. If immediately to the left of the verb is a relative conjunction or a comma, the previous clause is searched for a subject. If no subject is found to the left of the verb, a subject is searched for to the right of the verb.

If a subject is found, the next heuristic is forcing subject-verb agreement. Verb forms as well as nominal cases can be ambiguous. If a subject requires a verb in third person, tags denoting another person are deleted. Usually a subject will be in nominative unless a verb demands a different case. All other case tags will be deleted from the list of tags.

The 5^{th} heuristic marks objects of verbs. Direct objects as well as verb complements are marked as OBJ. An object is first looked for to the right of a verb and, if none is found, also to the left of the verb. Objects can be nominals or past participle verbs whereas the latter can only be complements. Verb complements are subject to the next heuristic. In Icelandic, adjectives adjust in case, number and gender to the noun. In case of verb complements this heuristic is forcing subject-object agreement.

In Icelandic, many verbs govern the case of their objects. Verb complements however are always in nominative. The 7^{th} heuristic forces this verb-object agreement. To make the decision about which case to use, a lookup table is used which is automatically derived from the IFD.

[4] SVO stands for Subject-Verb-Object. This refers to the order in which the parts of the sentences appear: First the subject, followed by the verb, followed by the object. The Romance languages, like French, follow SVO, as do English and the Scandinavian languages.

As mentioned above, nominals adjust in case, number and gender. To force agreement between nominals the heuristic starts looking for a nominal at the end of a clause and then searches towards the beginning. The head of a noun phrase in most cases is found on the right end of a noun phrase. When one is found it looks for modifiers to the left of the noun. All non-agreeing tags are removed from the noun and the modifiers.

Some prepositions in Icelandic can take two different cases. The heuristic first checks if one case tag can be removed by looking at the tags of the following word in the PP. If that does not bring an unambiguous result, verb-preposition pairs have to be looked at. A lookup table has been extracted from the IFD containing verb-preposition pairs with matching cases. If a match is found, agreement between the preposition and all other words in the PP is forced.

There are a few additional specific heuristics for choosing between supine and past participle, infinitive and active verb forms and ensuring feature agreement between reflexive pronouns and their antecedents. In case not all ambiguities have been removed, the default heuristic chooses the most frequent tag from the corpus. For more detailed examples on the heuristics see (Loftsson, 2006a).

2.2 Data-Driven Taggers

2.2.1 HMM-Based Taggers

2.2.1.1 TnT

Trigrams'n'Tags, or TnT for short, is a statistical Part-of-Speech tagger. It uses the Hidden Markov Model (HMM) which will be explained in reference to TnT in the next section. TnT is data-driven so it can be trained on virtually every tagset, under some restrictions (Brants, 2000). All it needs is a sufficiently large tagged training corpus.

It uses linear interpolation of unigrams, bigrams and trigrams on sparse data[5]. Suffix analysis is performed on unknown words. Therefore, a probability distribution is derived from all words in the training corpus. The length of the suffix[6] used depends on the word, but is at most 10 characters. Only words with low frequency are used

[5] Unigrams, bigrams and trigrams which do not occur in the corpus
[6] In this case a suffix is not a linguistic suffix but rather refers to the last x letters of a word

for suffix analysis of unknown words because unknown English words mostly are infrequent. This does not apply to Icelandic or German where new words can be built by concatenating two or more words. This can also be a drawback in languages which have a small training corpus where only a few words appear at low frequency. Capitalization is used to distinguish between proper nouns and other words but this does not work in languages like German either as every noun is capitalized.

The TnT tagger reaches an overall tagging accuracy of 96.7% for English, using the Penn Treebank and the same accuracy for German, using the NEGRA corpus, allthough the accuracy for known and unknown words differs slightly between the two languages (Brants, 2000). For more details on TnT, the reader is refered to (Brants, 2000).

2.2.1.2 Hidden Markov Model

The Hidden Markov Model (HMM) is used to find the probability of a sequence of events to occur. This can be used for taggers to predict the most likely tag for an ambiguous word. This section will give a short overview of HMM. A good introduction to Hidden Markov Models can be found in (Russell & Norvig, 1995)[7]. For a more detailed description, the reader is referred to (Manning & Schütze, 2002; Rabiner, 1989).

For building a Markov Model we observe events and use their probabilities to build a state transition matrix. This matrix can then be used to calculate the probability of any series of events. But for more complex, unobservable events we need to use a Hidden Markov Model. The transition matrix is no longer built from observation but from a stochastic process with an underlying stochastic process which is hidden.

Rabiner (1989) gives a demonstrative example. Imagine a person behind a curtain telling you the results of coin tossing but not telling you exactly what he is doing. You can only know the person is tossing coins but not whether its one or more coins nor in which order he is tossing them. This is one of the main problems of HMM, namely to identify the number of states the model should have. The more states, the more detailed it will be, but it also makes calculations more complicated and increases runtime. As an example we can take exactly the above coin tossing model. Using a 1-coin-model leaves us with 1 unknown parameter, a 2-coin-model leaves us with 4, a 3-coin-model with 9 unknown parameters and so on.

[7] (Russell & Norvig, 1995) pages 762-767

The idea of a Hidden Markov Model is exactly as the coin tossing example shows: we do not need to know the results of previous events. The only information we need is the current state to predict the next event. This is represented in the formula

$$P(X_{i+1}|X_i)$$

which gives the probability for event X_{i+1} following event X_i.

TnT uses four types of maximum likelihood probabilities:

$$\text{Unigrams: } \hat{P}(t_3) = \frac{f(t_3)}{N} \tag{2.1}$$

$$\text{Bigrams: } \hat{P}(t_3|t_2) = \frac{f(t_2, t_3)}{f(t_2)} \tag{2.2}$$

$$\text{Trigrams: } \hat{P}(t_3|t_1, t_2) = \frac{f(t_1, t_2, t_3)}{f(t_1, t_2)} \tag{2.3}$$

$$\text{Lexical: } \hat{P}(w_3|t_3) = \frac{f(w_3, t_3)}{f(t_3)} \tag{2.4}$$

The symbols used above denote the following: $f(t_i)$ is the frequency of a tag t_i, w_3 is the word in focus, N the total number of tokens in the training corpus and $\hat{P}$ is the maximum likelihood probability of the event. The probabilities shown above are the relative frequencies of a tag. In more detail, equation 2.1 gives the probability for a tag t_3 to occur. It is calculated by dividing the frequency of the tag t_3 by the number of tokens in the corpus. The same is done for Bigrams and Trigrams where the frequency for 2 or 3 tags respectively occurring in a row is calculated by dividing it by the frequency of the preceding tag(s) appearing in the corpus (for Trigrams appearing in that order). The lexical probability gives the probability of a word w_3 given a tag t_3. This is done by dividing the frequency of word w_3 appearing with tag t_3 divided by the frequency of tag t_3.

The probability for a sequence of words is calculated applying the maximum argument on the product of the contextual probability of a trigram and the lexical probability of the target word:

$$\arg\max_{t_1 \ldots t_T} P(t|w) = \prod_{i=1}^{T} P(t_i|t_{i-1}, t_{i-2}) P(w_i|t_i)$$

In TnT the resulting probability is then multiplied with a probability $P(t_{T+1}|t_T)$for an end-of-sequence marker for punctuations.

Because of data sparseness a smoothing algorithm is used. In HMM this is a linear interpolation of n-grams:

$$P(t_3|t_1,t_2) = \lambda_1\hat{P}(t_3) + \lambda_2\hat{P}(t_3|t_2) + \lambda_3\hat{P}(t_3|t_1,t_2)$$

The lambdas are estimated by deleted interpolation.

The unknown words are handled with suffix analysis. The probabilities for a given suffix are calculated from all words in the training corpus which have the same suffix. As a suffix, a fixed length of characters in the end of the word is evaluated. l_{n-j} refers to the j^{th} last suffix character of a n characters long word. The probabilities are again calculated with a maximum likelihood estimate:

$$\hat{P}(t|l_{n-i+1},...l_n) = \frac{f(t|l_{n-i+1},...l_n)}{f(l_{n-i+1},...l_n)}$$

Using suffixes of infrequent words showed better results for guessing unknown words. So words used for suffix handling have a frequency lower than a chosen threshold.

2.2.1.3 TriTagger

TriTagger is an extension of TnT. It uses the same list of idioms and the special lexicon for irregular verb forms that IceTagger is using. The gain in accuracy over TnT was rather small with only 0.02% (Loftsson, 2006b).

2.2.2 TreeTagger

The functionality of TreeTagger is similar to HMM based taggers. Instead of maximum likelihood estimations it uses decision trees to calculate the contextual probabilities. TreeTagger's lexicon consists of three parts: a fullform lexicon, a suffix lexicon and a default entry. All lexica are derived during training from an annotated corpus.

The fullform lexicon contains all words found during training together with their tag profiles with relative frequencies[8]. The suffix lexicon is derived from all words in the

[8] Tags with a relative frequency lower than 1% are not added.

corpus which belong to an open word class. It is organised as a tree. The search starts at an empty root, then goes backwards from the last letter of the word, to the second last and so on until a leaf is found containing a probability vector for the suffix. If a search in both lexica is unsuccessful, the default entry is chosen. For more details on the lexica, see (Schmid, 1994).

A binary decision tree is used to find the right tag. It is built recursively from a set of trigrams. The trigrams consist of the target word and the two preceding tags, tag_{-1} one word to the left and tag_{-2} two words to the left. A subtree in a decision tree can i.e. look like this:

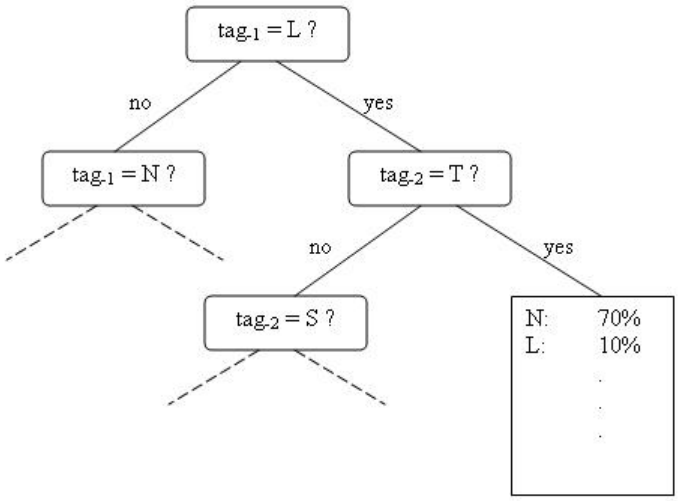

Figure 2.1: Binary decision tree for TreeTagger

Here the decision to be made is whether tag_{-1} is an adjective (L). If the answer is yes, TreeTagger chooses the branch to the right and explores tag_{-2}, otherwise it continues to the left branch of the tree and makes a new guess about tag_{-1}. This continues until a leaf is reaches which contains a hypothesis for the path taken. In this example the numeral (T) is chosen and the highest resulting probability from this path is $p(N|T, L)$ for a noun (N). The numbers in this example are randomly chosen. For more detail on TreeTagger see (Schmid, 1994).

2.2.3 Transformation-Based Learning

The main idea of transformation-based learning is to automatically learn rules that improve the current state of the training corpus. The tagger in training is using a

tagged and an untagged version of the corpus. It then runs through the following steps:

1. Initial class assignment: this can range from assigning random tags to using another tagger.
2. Compare to golden standard: now the output is compared to the hand annotated corpus.
3. Learning rules: from the errors found in the comparison, rules are created that can reduce the number of errors. The best rule is chosen, added to a list of rules and applied to the corpus. Only 1 rule is chosen in each cycle.
4. Return to step 2 until no rules can be found that make an improvement on the corpus (or the increase falls below a certain threshold).

The list of rules resulting from this algorithm is used as the set of rules in the tagger. More details on transformation-based tagging can be found in (Brill, 1995).

We used fnTBL (Ngai & Florian, 2001) in our experiment with combination of taggers (see chapter 2.4, 3.4.2.4 and 4.4 for more details). fnTBL is an improved version of TBL which works 13 to 139 times faster than the regular TBL. The extremely long running time of TBL is caused by the many iterations it has to take. It first tries out all the rules, applies one and then tries a lot of rules again, and so on. This is done because the change applied by one rule might effect a word or vicinity of a word which is affected by a succeeding rule. Instead, fnTBL stores the rules and in addition the set of samples on which the rule corrects the tag and the set of samples where the rule introduced an incorrect tag. It does not need to save those samples where no change is made when the rule is applied.

The fnTBL tagger reaches 96.61% tagging accuracy when trained on half of the Penn Treebank and 96.76% when trained on the whole corpus (Ngai & Florian, 2001). A more detailed description of fnTBL can be found in (Ngai & Florian, 2001).

2.2.4 Memory-Based Taggers

In memory-based tagging, an annotated training corpus is used to extract a lexicon and information about context. We used the MBT tagger (Daelemans, Zavrel, Berck, & Gillis, 1996; Zavrel & Daelemans, 1999) in our experiment.

In MBT, three data structures are automatically extracted: a lexicon, a known words case base and an unknown words case base. The lexicon contains each word of the

training corpus together with its tag profile and a frequency number for each of the tags calculated from its occurrences in the corpus. When untagged text is fed to MBT, every word is looked up in the lexicon. If a word is found, its information is retrieved from the lexicon. Additionally, its context is determined and its lexical pattern is looked up in the known words case base. The known words case base contains pattern of words and contexts from the training corpus. The obtained pattern is compared to patterns found in the case base and with a similarity metric (Daelemans et al., 1996) the nearest neighbor is evaluated. If a word is not in the lexicon, the unknown words case base is utilized. The last three characters of a word are evaluated for suffix analysis and the first letter for prefix analysis and capitalization of the word. The resulting pattern including the pattern of the context of the unknown word are then looked up in the unknown words case base. Again the best match is found by applying a similarity metric to calculate the nearest neighbor.

Daelemans et al. (1996) tested the MBT tagger on different languages and corpora. The best tagging results for English were achieved using the LOB corpus (97.0%), slightly lower accuracy when using the Penn Treebank (96.4%) and the highest for Spanish, using the CRATER Multi-Lingual Aligned Corpus (97.8%).

2.2.5 Maximum Entropy Taggers

We used the MXPOST maximum entropy tagger which is a Java based version of the PoS-tagger described in (Ratnaparkhi, 1996). It has a 5 word window, also called the history:

$$h_i = \{w_i, w_{i+1}, w_{i+2}, w_{i-1}, w_{i-2}, t_{i-1}, t_{i-2}\}$$

The probability of such a history is defined as

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)}$$

where π is a constant, and μ and $\alpha_{1,..,k}$ are chosen to maximize the joint probability of h and t. $f_{1,...,k}$ are features, that define this probability. Features are generated during training by scanning the training corpus with 'feature templates' (Ratnaparkhi, 1996) for each history h_i. Features ask yes/no questions about words and tags in the history h. An example of a feature is:

$$f_j(h_i, t_i) = \begin{cases} 1 & \text{if suffix } w_i = \text{leg and } t_i = \text{lvensf} \\ 0 & \text{otherwise} \end{cases}$$

This feature is looking for words ending on the suffix 'leg' and having the tag lvensf (adjective, feminine, singular, nominative, strong declension, positive). It is either active (1) or inactive (0), thus either contributing to the probability or not.

During tagging, the probability

$$p(t|h) = \frac{p(h,t)}{\sum_{t' \in T} p(h,t')}$$

is calculated for each history of a sentence and the highest N candidates (5 for MXPOST) of each history are maintained while evaluating the rest of the sentence. The probability for each candidate of the sentence $\{w_1...w_n\}$ is then calculated as:

$$P(t_1...t_n|w_1...w_n) = \prod_{i=1}^{n} p(t_i|h_i)$$

After the last word of the sentence has been added, the sequence with the highest probability is chosen. For more details on MXPOST see (Ratnaparkhi, 1996).

2.2.6 Bidirectional Tagger

One of the problems that occur in common tagging methods is that the tagger can not look back. It looks at one word at a time, in best case at a small window or uses a rule that takes the whole sentence into account. But once the routine has passed a word, it can not go back. Looking through a sentence from left to right or right to left is not necessarily the best solution. Sometimes we need to pick one word here, and one word there, in order to make a save decision.

A bidirectional tagger, a tagger based on a bidirectional sequence classification (Shen, Satta, & Joshi, 2007), applies such a method. For each sentence there are two sets:

- P - a set of accepted spans
- Q - a queue of candidate spans

The elements of the queue refer to spans which can be one word or a sequence of succeeding words in the sentence. The set P is initially empty. It will hold spans that have been assigned possible tags. A bidirectional tagger always keeps a number

of hypotheses for each span. In the experiment in (Shen et al., 2007) the number of choices kept for each tag (also referred to as beam width) is 3. I will use this test setup to explain the functionality of a bidirectional tagger.

The algorithm picks one element of Q at a time. The element is not picked in order of appearances but by its probability. If one word in the sentence, for example, has only one tag, this tag is unambiguous and the word will be chosen as a starting point. Every assigned tag influences the choice of possible tags in its context, thus the other words of the sentence. It minimizes the choice of ambiguous tags in its surrounding. Due to this minimization, a beam width as small as 2 or 3 is enough to get competitive results.

The general algorithm works as follows: the word or span in Q which has the tag with the highest probability is picked, the three most favorable tags are assigned to it and it is added to the spans in P. If there are spans in P with adjacent words, the new span is joined with these spans. Otherwise it is added as a new span. This is repeated until all words from Q have been added to P which then contains one span for the whole sentence with three different hypotheses. The hypothesis with the highest probability is then chosen.

Shen et al. (2007) tested their algorithm with different feature sets on the Penn Treebank. The best system had an error rate of 2.67%.

2.2.6.1 BI+WC+CT

The bidirectional tagger used in chapter 4.4 is called *BI+WC+CT*[9]. It is an enhanced version of the bidirectional tagger described above and was developed for Icelandic by Dredze and Wallenberg (2008).

Because of the large Icelandic tagset and the long training time required by the bidirectional tagger, Dredze and Wallenberg divided the tagset into 11 separate learning problems, one for each word class. In this way the tagger only needs to evaluate tags of one word class which reduces the complexity of the tagset. This reduced the tagging time from 4 days to 12 hours and increased the tagging accuracy[10]. This step was called *WC* (Word Class).

[9] I will refer to this tagger as BI in the tables in chapter 4.4

[10] The numbers for this experiment are not comparable with other results given here, because it was only evaluated on one of the 10 test sets due to the long training time. Further details can be found in (Dredze & Wallenberg, 2008).

Most tagging errors that occurred at this point can be attributed to case errors in nouns, adjectives and pronouns. Dredze and Wallenberg added a Case Tagger (CT) which takes a fully tagged sentence as an input and retags the case and gender of nouns, adjectives and pronouns using long dependencies to take the context of the whole sentence into account. A full description of the features chosen for this tagger can be found in (Dredze & Wallenberg, 2008).

2.3 Integrated Taggers

When integrating a tagger, its features are used in a second tagger. Loftsson (2006b) used several integrated taggers, described in the following section.

2.3.1 fnTBL*

Due to the low tagging accuracy of fnTBL for unknown words, Loftsson (2006b) integrated IceMorphy to provide fnTBL with an initial tag. The tag with the highest frequency was assigned to each unknown word before fnTBL applies its rules. This increased the overall tagging accuracy from 89.33% to 90.15%.

2.3.2 TnT*

IceMorphy and its gap filling property was used to generate a ”filled” lexicon. The missing tags in the lexicon that TnT is generating are filled by IceMorphy. Because TnT’s lexicon requires frequency information, the new tags are assigned a frequency of 1. The overall tagging accuracy went up from 90.44% to 91.18% (Loftsson, 2006b).

2.3.3 Ice+HMM

As described in section 2.1.2.0.3, IceTagger is using a default heuristic in case a word can not be fully disambiguated. This default heuristic is simply choosing the tag with the highest frequency. To improve the accuracy for those words, TriTagger was called from within IceTagger for full disambiguation. The overall tagging accuracy was increased from 91.54% to 91.80% (Loftsson, 2006b). The TriTagger, as mentioned

above, is an extension of the HMM based TnT tagger. Therefore we refer to this tagger as Ice+HMM.

2.3.4 HMM+Ice

HMM+Ice (Loftsson, Kramarczyk, Helgadóttir, & Rögnvaldsson, 2009), as opposed to Ice+HMM is using TriTagger before starting disambiguation and not as the last step in disambiguation as done in Ice+HMM. HMM+Ice starts with IceTagger looking up tag profiles for known words and gap filling with IceMorphy. Then a copy is made of the generated tag profiles and TriTagger disambiguates the tags in this copy. IceTagger then uses the tagging results of TriTagger to eliminate tags from the tag profiles which do not belong to the same word class as the tag chosen by TriTagger. IceTagger then runs its disambiguation on the remaining tags.

2.4 Tagger Combinations

Different taggers have different advantages and disadvantages. One makes more tagging errors than another tagger in one field but gives better results in another field. By combining more taggers, these errors can be minimized. Results presented by Halteren, Daelemans, and Zavrel (2001) show that a combination of taggers can score a higher accuracy than each individual tagger.

There are several approaches to decide between the tags suggested by the taggers. In our experiments we used simple and weighted voting as well as linguistically motivated rules. For more details on combination and voting methods, the reader is referred to (Halteren et al., 2001).

2.4.1 Simple Voting

In simple voting, every tagger gets one vote. The tag with the most votes is then chosen. This can lead to problems in case of a tie. To avoid this a default decision or weighted voting can be used.

2.4.2 Weighted Voting

In weighted voting, taggers which give better results get a higher vote. They are only outvoted if many other taggers agree on a different tag. Finding the right weights can be difficult. Giving one tagger a weight that is too high, can exclude all others from decision making. Giving the best tagger a weight that is too low, can cause it to be outvoted by worse taggers, causing a wrong tag to be chosen.

2.4.3 Linguistically Motivated Rules

A linguistically motivated rule (LMR) is used if taggers have advantages under strictly defined circumstances. The combination of taggers, using different tagging methods, gives us the possibility of choosing taggers of one method that yields better results in one situation and choosing other taggers in a different situation.

As an example, TnT is restricted by a small window size and can not take long dependencies (dependencies reaching over a whole sentence) into account whereas IceTagger has the advantage of its heuristics and can handle these situations better than TnT. An example of LMRs in use will be given in chapter 3.4.2.4.

Chapter 3

Related Work

3.1 Accomplished Goals in Icelandic Language Technology

As mentioned in the introduction, a number of goals were set in the original report about the status of Language Technology in Iceland (Ólafsson et al., 1999). Some of them have been met, others are still pending. The following section will give a short overview of the status of different fields in which PoS tagging plays a role.

3.1.1 IceNLP

IceNLP is the tagging and parsing system for Icelandic. It has been developed by Loftsson and Rögnvaldsson (2007a). IceNLP as it is today consists of the following parts [1]:

1. Preprocessor
   (a) Sentence segmentation
   (b) Tokenisation
2. PoS tagger - IceTagger
   (a) Morphological analyzer - IceMorphy
      i. Lexicon lookup

[1] Borrowed from (Loftsson & Rögnvaldsson, 2007a)

ii. Unknown word guessing

iii. Tag profile gap filling

(b) Disambiguate

i. Local rules

ii. Heuristics

3. Finite-state parser - IceParser

(a) Phrase structure module

(b) Syntactic function module

3.1.1.0.1 Preprocessor

As mentioned before, the task of the Preprocessor is to prepare the input text for IceTagger. It carries out tokenisation, which means breaking the text down into its basic tokens (words, numbers and punctuation marks). If necessary sentence segmentation is carried out additionally in case its unclear where one sentence ends and another one starts.

3.1.1.0.2 IceTagger

IceTagger is a linguistic rule-based tagger especially developed for Icelandic. It reads in continuous text and analyses it. It assigns to each word a tag that contains grammatical information about this word. IceTagger was described in more detail in chapter 2.1.2.

3.1.1.0.3 IceParser

IceParser is an incremental finite-state parser. It consists of 22 transducers stringed together, where each transducer adds more information to the text and the preceding transducers output is the succeeding transducers input. The parser consists of two modules:

- phrase structure module (14 transducers): This module adds two groups of phrasal labels: main labels NP, VP, AP, PP and AdvP (Noun Phrase, Verb

Phrase, Adjective Phrase, Prepositional Phrase and Adverbial Phrase) and some additional labels CP, SCP, InjP and MWE (coordinating conjunction, subordinate conjunction, interjection and multiword expression). APs and NPs mark a sequence of Adjective and Noun Phrases respectively.

- syntactic functions module (8 transducers): This module adds the syntactic labels *QUAL, *SUBJ, *OBJ, *OBJAP, *OBJNOM, *IOBJ, *COMP and *TIMEX (genitive qualifier, subject, object, object of an AP, nominative object, indirect object, complement and temporal expression).

IceParser takes the output of IceTagger as an input, i.e. tagged text. Although the tags would give a lot of useful information, the parser only uses the word class information and the grammatical case information. The reason given by the authors in (Loftsson & Rögnvaldsson, 2007b) is that IceParser should be used as a grammatical checker among other things. If it would e.g. make use of feature agreement it could not find grammatical errors because it would not count words to a phrase whose features (case, gender, ...) do not agree with the rest of the phrase.

Every phrase consists of a beginning and an end label. Phrase structures are indicated with square brackets and syntactic functions with curly brackets:

{*SUBJ> [NP vagnstjórinn NP]*SUBJ>} [VP sá VP]
{*OBJ< [NP mig NP] *OBJ<}
('The driver saw me')

In this example from the IFD, the noun vagnstjórinn ('the driver') is part of a noun phrase. The noun phrase is also part of the subject which is pointing towards the verb phrase sá ('saw'). On the other side of the verb phrase, the object is again pointing towards the verb.

The transducers are sorted starting from the deepest constituent. To illustrate this, consider an example from (Loftsson & Rögnvaldsson, 2007b). The regular expression used to identify adverbial phrases is as follows:

$$Adv = \{WordSpaces\}\ \{AdvTag\}$$

WordSpaces can be any sequence of characters excluding spaces. This regular expression marks any word that stands before an adverb PoS-tag as an adverbial phrase. A regular expression used to identify an adjective phrase would look like this:

$$Adj = \{WordSpaces\}\ \{AdjTag\}$$
$$AdjPhrase = \{AdvPhrase\}?\{Adj\}$$

The optional adverbial phrase in the regular expression can only be found if it has been assigned before the transducer for adjectives is called. As mentioned in the beginning, IceParser is an incremental finite-state parser. It can not jump back to a previously run routine. So all the information required in a regular expression needs to be available at the time it is applied. The transducers can be as simple as one state (adverb) or as complicated as 50.000 states (noun phrase).

3.1.2 Lemmatizer

Lemmald, a lemmatizer for Icelandic, was recently developed by Ingason, Helgadóttir, Loftsson, and Rögnvaldsson (2008). There has been a language-independent CTS lemmatizer available that was trained on Icelandic, but Lemmald is the first one especially developed for this language. It uses a mixed method approach, combining data-driven methods with linguistic approaches. Lemmald makes use of three resources: IceTagger, the IFD corpus and the lexical database BÍN. The IFD corpus is used for training and BÍN can be used optionally as a backup dictionary. Lemmald processes input in this manner: $getLemma(wordForm, tag)$. The pair of a word form and a tag is unambiguous in most cases. In order to get this pair, the input text has to be tagged. This is done by using IceTagger. Lemmald relies on the accuracy of this tag. It uses the information to determine the morphological group the word is appointed to. IceTagger only has a tagging accuracy of 91.59% which is not high enough for this purpose. On the other hand Lemmald uses only one piece of the assigned tag, the word class. When considering only this piece of information, IceTagger reaches an accuracy of 97.61% (Loftsson et al., 2009) which is considerably higher than the accuracy that includes the full depth of tagging information.

We used the lemmatizer in the third part of our experiment, which will be discussed in chapter 4.3. A detailed description of Lemmald can be found in (Ingason et al., 2008).

3.1.3 Applications

Tagging and parsing are the basis for most applications in language technology. I will give a few examples on applications that have been developed for Icelandic.

A few simple spell checkers have been developed for Icelandic over the years such as Púki[2], the dictionary Tölvuorðabók[3], which would work as a spell checker, the spell checker developed by the Dutch company Polderland[4], or the spell checker by Aspell[5] which is an open source spell checker for GNU/Linux supporting Icelandic. All of those spell checkers have one thing in common: they check one word at a time and do not take the context into account. In many cases a typing error can lead to another word which also exists but is of the wrong type or in the wrong case to be in that position. To identify and correct this kind of error, a context-sensitive spell checker is needed. It needs to analyze the text first and then find all possible words that could come instead the faulty word. No commercial spell checker of this type exists yet for Icelandic but Ingason, Jóhannsson, Helgadóttir, Loftsson, and Rögnvaldsson (2009) have made a first attempt to use three different methods for context-sensitive spelling correction which have worked well for English. Tagging of the input text is done with IceTagger. The results are, as expected, lower than for English but still close to 90% accuracy. Data sparseness and errors in the tagging results were named as the main reason for the the lower accuracy.

Not much has been done yet in the field of machine translation. The simplest way to translate a text between two languages is a word-by-word translation. That might give understandable results in related languages but will cause problems in languages with different word order. To be able to change the word order according to the syntax of the languages it requires a tagger and parser as an underlying system.

A few translation tools are offered on the Internet, such as InterTran[6] or Stefán Briems Tungutorg[7], but all in all the results are poor. The translation of the simplest sentences such as ”Ég vil fisk” ('I want fish') fails at the word vil ('want') which is translated as ”will” by Tungutorg and is not even part of the dictionary of InterTran. The latter is also lacking the word fish. Nonetheless, the translation of a news article is more or less understandable in Tungutorg. This supports the assumption that the main issue is not the underlying program rather the lack of a good dictionary. A project on machine translation is in progress right now at Reykjavík University.

Even speech systems rely on tagging and parsing in order to work properly. To find the intonation of a sentence, syntax analysis is necessary. The same applies to speech recognition, when analyzing a sentence.

[2] For more information see http://vefur.puki.is
[3] http://www.alnet.is
[4] http://www.polderland.nl
[5] http://aspell.net
[6] http://www.tranexp.com
[7] http://www.tungutorg.is

During 1989-1993 a speech simulator from the Swedish company Infovox[8] was adapted for Icelandic by the Institute of Linguistics and the Faculty of Engineering of the University of Iceland and the Icelandic Federation of the Handicapped.

The latest speech simulation program for Icelandic is Vefþulan[9]. It was developed by the University of Iceland, Iceland Telecom and Hex Software and trained by Nuance. Hex Software first developed the voice of Ragga, the female speaker, from 2005 until spring 2006. Ragga´s voice is only used in Vefþulan but can also be used in other applications. Though Hex Software does not plan on developing more applications of that kind, the voice is available for interested companies. Since February 2008, Vefþulan has been available as a web interface. It can be integrated into web pages as a service, as the paper Morgunblaðið did in June 2008.

3.2 The Icelandic Tagset

The Icelandic tagset has been developed along with the construction of the IFD (Pind et al., 1991). It consists of 700 possible tags but only 639 of them actually occur in the IFD. The tags are very detailed. There are 8 main types of words. Each of them has their specific list of attributes (see Table 3.1)

If we look at an example sentence "*Það er gott veður í dag*" ('It is good weather today'), IceTagger returns the following line:

Það fphen er sfg3en gott lhensf veður nhen í ao dag nkeo . .

This sentence gives an overview of the five word groups. As an example of a noun we can take *veður* ('weather'). It gets the tag *nhen* which stands for **n**afnorð ('noun'), **h**vorugkyn ('neuter'), **e**intala ('singular'), **n**efnifall ('nominative'). As an example of a verb we look at *er* ('is'). The tag *sfg3en* stands for **s**agnorð ('verb'), **f**ramsöguháttur ('indicative'), **g**ermynd ('active voice'), 3^{rd} person (3), **e**intala ('singular'), **n**útíð ('present tense'). An adjective is *gott* ('good') with the tag *lhensf*: **l**ýsingarorð ('adjective'), **h**vorugkyn ('neuter'), **e**intala ('singular'), **n**efnifall ('nominative'), **s**terk beyging ('strong declension'), **f**rumstig ('positive'). The personal pronoun *það* ('it') has the tag *fphen* which stands for **f**ornafn ('pronoun'), **p**ersónufornafn ('personal pronoun'), **h**vorugkyn ('neuter'), **e**intala ('singular'), **n**efnifall ('nominative'). The preposition *í* ('in') gives us information about which case it governs. The tag *ao* stands for forsetning ('preposition') (a), þ**o**lfall ('accusative') which means that it

[8] http://www.orin.com/access/infovox_ivox/index.htm
[9] A free user interface is available at http://www.hexia.net/upplestur

type of word	**attributes**	**possible tags**
noun	gender, number, case, article, proper noun	224
adjective	gender, number, case, declension, degree	144
pronoun	subcategory, gender/person, number, case	184
article	gender, number, case	24
numeral	category, gender, number, case	27
verb (exc. past participle)	mood, voice, person, number, tense	59
verb (past participle)	mood, voice, gender, number, case	24
adverb and preposition	category, degree	9
conjunction	sign of infinitive, relative conjunction	3
foreign word		1
unanalyzed word		1
sum		700

Table 3.1: The Icelandic tagset

governs the accusative in its prepositional phrase. This can be seen in the word *dag* ('day') which stands in accusative (in *nkeo* the o stands for accusative). A complete table of the Icelandic tagset is provided in the Appendix.

3.3 Dictionaries and Corpora

The only PoS tagged corpus available for Icelandic is the IFD corpus. It is a carefully balanced corpus consisting of 590,297 running words, including punctuations and numerals. It has been put together from 100 fragments, 5,000 words each, from a variety of publications, published between 1980 and 1989. The texts are evenly distributed amongst the genres of Icelandic fiction, translated fiction, biographies and memoirs, non-fiction and Icelandic and foreign books for children and youngsters. There are no two books included that were written by the same author or translated by the same person.

Although this sounds like a suitable corpus, there are many disadvantages. First, the size of only half a million running words is fairly small compared to tagged corpora in other languages which often contain millions of words. The Penn Treebank for

English e.g. contains 7 million words of Part-of-Speech tagged text (Taylor, Marcus, & Santorini, 2003). Furthermore, the IFD contains only 59,358 word forms, including punctuation and numerals. 15.9% of the word forms also turned out to be ambiguous (Helgadóttir, 2005). This high number can be explained through the richness of Icelandic inflections. A noun has up to 16 tags, adjectives have up to 120 tags and verbs have up to 106 tags (numbers taken from (Bjarnadóttir, 2004)).

An example is the word form *á*. It can be a preposition which governs either the accusative, the dative or no case (3 tags), it can be an interjection, it can be the feminine noun denoting a river (3 tags: nominative, dative and accusative), it can be the dative and accusative of the word *ær* 'ewe' (2 tags) and it can be the 1^{st} and 3^{rd} person singular present tense of the verb *eiga* 'to own' which gives another two tags. This gives a total of 11 tags which is not even close to the highest possible number. The word form *minni* can have over 30 different tags originating from the words *lítill* 'small', *minn* 'my', *minna* 'to remind' and *minni* 'memory'.

Two points that have already been criticized in (Ólafsson et al., 1999) are that the IFD does not contain any samples of spoken language and that the text samples used were already a decade old in 1999. A corpus has to be updated regularly and this has not been done since.

A lexical database that is available for Icelandic is BÍN, Inflections of modern Icelandic[10] (Beygingarlýsing íslensks nútímamáls) (Bjarnadóttir, 2005). It contains 270,000 words with over 5.8 million word forms[11]. The words are mostly from modern Icelandic, personal names and proper nouns.

What is still missing to fulfill the requirements of a BLARK is a bilingual or multilingual dictionary for machine translation. A Nordic dictionary[12] is in progress, which is supposed to contain 50,000 words and translations to Norwegian, Swedish and Danish. The dictionary will only be available online. It will be linked to the Icelandic Inflections Project (BÍN) and therefore offer full inflection tables on all the words in the dictionary. In addition it will contain over 6,000 common phrases. The development is still in progress.

[10] Also referred to as the comprehensive Morphological Database of Icelandic Inflections (MDII).

[11] The latest status can be found on (Bjarnadóttir, 16. November 2007). According to this source, it has been increased to 8 million word forms.

[12] More information can be found in Icelandic at http://www.arnastofnun.is/page/arnastofnun_ord_islex

3.4 Accuracy in Icelandic Tagging

3.4.1 Evaluation Method

To evaluate the accuracy of different taggers for Icelandic, 10-fold cross-validation has been used. Helgadóttir (2005) prepared the data for this method. The Icelandic Frequency Dictionary was split into 10 parts and 10 test sets were created from them. The first test set uses parts 2 through 10 for training and part 1 for tagging, the second test set is trained on parts 1 and 3 through 10 and is used for tagging part 2, and so on. An example of the output of such a test is shown in the table below.

	u.w.r.	**unkn.**	**known**	**overall**
01	7.52%	75.02%	92.81%	91.47%
02	6.73%	75.16%	92.66%	91.48%
03	6.81%	74.20%	92.81%	91.54%
04	6.64%	75.85%	92.68%	91.56%
05	6.46%	76.23%	93.02%	91.94%
06	6.67%	75.94%	92.88%	91.75%
07	6.65%	73.96%	92.89%	91.63%
08	6.90%	75.45%	92.70%	91.51%
09	6.71%	75.86%	92.55%	91.43%
avg	**6.79%**	**75.30%**	**92.78%**	**91.59%**
10	6.75%	80.69%	93.75%	92.87%

Table 3.2: This table presents the full tagging results of IceTagger with its 10 tag sets and the average numbers over the first 9 tag sets

The *unknown word ratio* (u.w.r.) denotes the ratio of unknown words relative to the number of words in the text. The columns *unknown words* (unkn.) and *known words* show the tagging accuracy for the unknown and known words respectively. The *overall tagging accuracy* (overall) denotes the combined accuracy of known and unknown words over the whole text. The line marked with *avg* shows the average numbers over the first 9 test sets. The results presented in chapter 4 show these average numbers. Set nr. 10 was used for development and does not count as a reference result. It is only used for evaluation and presented here for the sake of completeness.

3.4.2 Previous work on Tagging Icelandic Text

3.4.2.1 Linguistic Rule-Based Taggers

IceTagger is the only linguistic rule-based tagger for Icelandic. The tagging results are presented in table 3.2.

3.4.2.2 Data-Driven Taggers

Helgadóttir (2005) made an experiment with three different data-driven taggers. She used TnT, using the Hidden Markov Model (see 2.2.1.2), MXPOST, a maximum entropy tagger (see 2.2.5) and fnTBL for transformation-based tagging (see 2.2.3).

Tagging other languages has shown a correlation between the size of a tagset and the tagging accuracy (Loftsson, 2008b). The bigger the tagset the lower accuracy is achieved. Therefore, Helgadóttir made an attempt to simplify the tagset and got a significant improvement over the old numbers (see table 3.3). In the simplified tagset, classification of adverbs and conjunctions, and subcategorization of pronouns were ignored. The highest improvement was reached when considering only the word class of a word (11 tags instead of 700) but this simplification is only usable in a few applications.

Another attempt was to reduce the number of unknown words. Unknown words make up close to 7% of the words and reached tagging accuracies between 54.03% in fnTBL and 71.60% in TnT. A backup lexicon was produced which contained approximately half of the unknown words in each test set. Only two of the taggers supported the use of such a lexicon before the unknown word handler. The results of all the above described tests are shown in table 3.3. The row *Unchanged* shows the tagging accuracy of the taggers trained on the IFD without applying any changes.

	MXPOST	fnTBL	TnT
Unchanged	89.08%	88.80%	90.36%
Simplified Tags	90.46%	89.98%	91.83%
Backup Lexicon	N.A.	90.06%	91.54%
Only Word Class	97.29%	97.25%	98.14%

Table 3.3: Tagging results for three different data-driven taggers

Other experiments with data-driven taggers have been made in recent years. Loftsson (2006b) used MXPOST, MBT, fnTBL, TnT and TriTagger which will be described in more detail in the following section. Henrich, Reuter, and Loftsson (2009) used the TreeTagger in addition. The results of their experiment will be presented in section 3.4.2.4. BI+WC+CT was first used in Dredze and Wallenberg (2008). The experiment will be described in more detail in chapter 4.4. The tagging results for all data-driven taggers that have been used[13] can be found in table 3.4.

	unkn.	known	overall
MXPOST	62.29%	91.00%	89.03%
MBT	59.40%	91.47%	89.28%
TreeTagger	61.75%	91.28%	89.30%
fnTBL	55.51%	91.82%	89.33%
TnT	71.68%	91.82%	90.44%
TriTagger	71.04%	91.87%	90.46%
BI+WC+CT	69.74%	93.70%	92.06%

Table 3.4: Tagging results for all data-driven taggers

3.4.2.3 Integration of Taggers

Integration of taggers means using a feature or functionality of one tagger in the tagging process of another tagger. Loftsson (2006b) describes several integration methods used for tagging Icelandic (see also chapter 2.3). In the first one, IceMorphy provided fnTBL with an initial tag for unknown words. The second method was to generate a ”filled” lexicon for TnT where IceMorphy is generating the missing tags in a tag profile and adds frequency 1 to the new tags (for gap filling see 2.1.2.0.1). The third attempt was to call IceMorphy from within TriTagger. TriTagger is an extended version of TnT where a list of idioms and a special lexicon containing irregular verb forms usually used by IceTagger are added as a backup lexicon. In the last one, IceTagger calls TriTagger for full disambiguation instead of using its default heuristics (see chapter 2.1.2.0.3). The results for the integration of taggers can be seen in table 3.5 (the upgraded versions are marked with a star).

[13] The overall tagging accuracy for TreeTagger was published in (Henrich et al., 2009). The accuracies for known and unknown words have not been published yet. Hrafn Loftsson provided me with these numbers.

	unkn.	**known**	**overall**
fnTBL	55.51%	91.82%	89.33%
fnTBL*	66.30%	91.90%	90.15%
TnT	71.68%	91.82%	90.44%
TnT*	72.80%	92.54%	91.18%
TriTagger	71.04%	91.87%	90.46%
TriTagger*	74.46%	92.58%	91.34%
IceTagger	75.09%	92.74%	91.54%
IceTagger*	75.33%	93.00%	91.80%

Table 3.5: Tagging results for three integrated taggers

3.4.2.4 Combination of Taggers

In the test series in (Helgadóttir, 2005) there was a first attempt made to combine taggers. Simple voting was applied to the three taggers used, MXPOST, fnTBL and TnT. Each tagger had equally weighted votes, but in case of a tie the tag produced by the best tagger, in this case TnT, was chosen. The precision improved from an overall 90.36% from TnT as the strongest tagger to 91.54% for the combined tagger.

Helgadóttir also tested the influence of the literary genre on the tagging results. She used texts between 2,700 and 6,000 words from the genres literature from the late 19^{th} to the early 20^{th} century, literature after 1980, technical literature from the field of computer science and information technology and a text about law and business. The text pieces were tagged by all three taggers which were trained on the whole IFD. The results for these texts can be found in table 3.6:

genre	accuracy
late 19th, early 20th century literature	93.93%
literature after 1980	92.91%
computers and information technology	86.77%
law and business	88.54%

Table 3.6: Tagging results from corpora of different genres, tagged by a combination of three taggers.

This proves that the IFD is strongly based on literature and needs a wider spread if it should be used on any kind of text in the future.

In (Loftsson, 2006b) this experiment was continued with more taggers. The author added MBT and IceTagger and beside that he also used the integrated taggers described above. He used simple voting as well as LMRs (Linguistically motivated rules). LMRs are used in cases where Data-Driven taggers are more likely to give a wrong result whereas IceTagger is more likely to suggest the right tag but is outvoted by the other taggers. Two such rules were applied:

1. In cases of long dependencies IceTagger finds the correct tag where data-driven taggers make more mistakes due to their limited context windows size. Examples are long dependencies between a subject and a verb in 1^{st} person or long dependencies between a subject and a reflexive pronoun. In these cases the IceTagger's tag will always be selected and the votes of the other taggers will be ignored.

2. The second rule is a feature agreement constraint: "If all the tags, provided by the individual taggers for the current word, are nominal tags and the current tag provided by Ice agrees in gender, number and case with the preceding (selected) nominal tag or the following (yet to be selected) nominal tag, then choose Ice's tag" ((Loftsson, 2006b), chapter 6).

The results for this experiment are shown in table 3.7.

	unkn.	**known**	**overall**
MXPOST + fnTBL + TnT	71.80%	92.99%	91.54%
fnTBL + TnT + Ice	76.76%	93.77%	92.61%
fnTBL* + TnT* + Ice	76.55%	94.13%	92.94%
MXPOST + MBT + fnTBL + TnT + Ice	76.74%	93.97%	92.80%
MXPOST + MBT + fnTBL* + TnT* + Ice	78.70%	94.36%	93.29%
MXPOST + MBT + fnTBL* + TnT* + Ice*	78.65%	94.41%	93.34%
MXPOST + MBT + fnTBL* + TnT* + Ice* + Rule 1	78.66%	94.50%	93.43%
MXPOST + MBT + fnTBL* + TnT* + Ice* + Rule 1 & 2	78.68%	94.56%	93.48%

Table 3.7: Results for combination of five taggers

As can be seen here, combinations of only 3 taggers can give a significant increase of tagging accuracy of over 1% compared to IceTagger and almost 1.5% if the improved versions of the taggers (marked with *) are used. Combining the improved versions of fnTBL, TnT and IceTagger with the MXPOST and MBT tagger increased the

accuracy by almost 2%. Adding the rules did not result in a significant change but still improved the accuracy by another 0.14%.

The most recent experiment for combinations of taggers tested on Icelandic was done by Henrich et al. (2009). The authors developed a tool called CombiTagger which can simulate any combination of taggers with simple or weighted voting. It takes output files from taggers as an input and combines them according to the order the user suggested. Simple voting assigns every tagger a vote of equal weight and in case of a tie chooses the tag of the tagger that was rated the highest by the user. In weighted voting the user can himself choose weights for each tagger. CombiTagger's output then provides statistics about the agreement between the taggers, agreement of the taggers with the golden standard and tagging accuracies. It also provides a table which offers highlighting for the same statistics. In addition, tags of the combined output and the golden standard can be changed. For more information about CombiTagger, the reader is referred to (Henrich et al., 2009).

Henrich et al. used six taggers for their experiments: IceTagger, TnT, fnTBL, MBT, MXPOST and TreeTagger. In addition they used the same improved versions of IceTagger, TnT and fnTBL (marked with *) as Loftsson (2006b).

	voting method	overall
fnTBL*, Ice*, MBT, MXP, TnT*	simple	93.09%
Ice*, TnT*, fnTBL*, MBT, MXP	simple	93.35%
Ice*, TnT*, fnTBL*, Tree, MBT, MXP	simple	93.24%
fnTBL*, Ice*, MBT, MXP, TnT*	weighted	93.33%
Ice*, TnT*, fnTBL*, MBT, MXP	weighted	93.41%

Table 3.8: Results from CombiTagger

The highest result reached with simple voting was 93.35% with 5 taggers. TreeTagger reduced the accuracy as can be seen in line 3 so it was left out from further tests. The highest accuracy reached for weighted voting was 93.41%, using the same combination of taggers as in simple voting, which is an increase of 0.06%.

One of the statistics offered by CombiTagger is the percentage of words which have not been tagged correctly by any tagger. It was 2.29% for 6 taggers which means that the maximum tagging accuracy that can be reached by these taggers is 97.71%.

Chapter 4

Experiment

As mentioned before, the tagging accuracy for Icelandic, using a single tagger, is relatively low compared to other languages. It has been as high as 96-97% for English and German (Brants, 2000) and still 93.55% for Swedish (Megyesi, 2002) which is closer related to Icelandic than the other two languages, but only 91.54% for Icelandic (Loftsson, 2006b). The highest result reached for Icelandic was through a combination of five taggers which gave a tagging accuracy of 93.48% (Loftsson, 2006b)

In the following four sections, I will explain in detail what approaches we used to improve the results for Icelandic. It should be noted that the experiments have not been made in the same order they are reported in. The work on this project started in September 2007 with an older version of IceTagger. All tests done in the beginning of the project had to be redone with the newer version[1]. Furthermore, improvements reached from later experimental setups were combined with and tested on older setups to make sure that we can get the highest improvement possible for tagging Icelandic text.

4.1 Using a Larger Dictionary

Although the accuracy reached for known words in IceTagger is as high as 92.78% (see table 3.2) at average, it is considerably decreased by the low accuracy for unknown words of 75.30%, which results in only 91.59% overall tagging accuracy. The unknown words have a ratio of 6.79% on average which has a big influence on the overall

[1] The new version of IceTagger reaches an overall tagging accuracy of 91.59% (Loftsson et al., 2009).

results. We therefore tested the influence of the size of the dictionary on the tagging accuracy of IceTagger and TnT[2]. A flat file was extracted from the Database of Modern Icelandic Inflections (BÍN, see chapter 3.3) which resulted in over 6 million entries. We extracted information from this file and added it to the dictionaries that IceTagger and TnT, respectively, extracted from the IFD.

We did not use all 6 million entries because many of them contained information the taggers could not use. Some information even produced more errors. We tried the following options to filter useful information from BÍN:

a) all entries

b) every word that is not a noun

c) proper and common nouns with or without article

d) all adjectives

e) infinitive, present participle and supine verbs in active or middle voice

f) indicative and subjunctive verbs

g) past participle active verbs in nominative

h) adverbs

In the end the highest accuracy was reached with a combination of c, d, e, f, g and h.

The entries in the extracted file are in a format of one word and one tag per line. Each word can have several different tags and will therefore appear in several lines with different tags. Each line contains extra information such as the lemma of the word, which we do not use. The tags used were not consistent with the IFD's tagset. Therefore we not only had to extract the words from BÍN but, in addition, change the tags according to our tagset. An example from BÍN for the word *tannkrem* ('tooth paste') is shown in figure 4.1. Each entry consists of the lemma of the word, a running number of the lemma (not of the word form), an abbreviation indicating the morphological class, the word form and a tag from a different tagset. This tag had to be changed to the tagset used in the IFD.

The abbreviation determines which subroutine is invoked. A noun is marked either with *kk* (masculine), *kvk* (feminine) or *hk* (neuter). Adjectives are marked *lo*, verbs

[2] This part was a joint work with Hrafn Loftsson.

tannkrem;41792;hk;alm;tannkrem;NFET;
tannkrem;41792;hk;alm;tannkremið;NFETgr;
tannkrem;41792;hk;alm;tannkrem;ÞFET;
tannkrem;41792;hk;alm;tannkremið;ÞFETgr;
tannkrem;41792;hk;alm;tannkremi;ÞGFET;
tannkrem;41792;hk;alm;tannkreminu;ÞGFETgr;
tannkrem;41792;hk;alm;tannkrems;EFET;
tannkrem;41792;hk;alm;tannkremsins;EFETgr;
tannkrem;41792;hk;alm;tannkrem;NFFT;
tannkrem;41792;hk;alm;tannkremin;NFFTgr;
tannkrem;41792;hk;alm;tannkrem;ÞFFT;
tannkrem;41792;hk;alm;tannkremin;ÞFFTgr;
tannkrem;41792;hk;alm;tannkremum;ÞGFFT;
tannkrem;41792;hk;alm;tannkremunum;ÞGFFTgr;
tannkrem;41792;hk;alm;tannkrema;EFFT;
tannkrem;41792;hk;alm;tannkremanna;EFFTgr;

Figure 4.1: Entries for the word *tannkrem* ('toothpaste') from the Database of Modern Icelandic Inflections

are marked *so* and adverbs *ao*. The appropriate subroutine then parses and alters the tag from the BÍN tag to the corresponding IFD tag.

As an example, we can look at the tag of line 2 in figure 4.1 *NFETgr*. We already know from the abbreviation, that tannkremið is a neuter noun (hk) which will be replaced by *n* for noun and *h* for the gender neuter. The first two letters of a noun tag denote the case. *NF* is replaced by *n* (nominative). The next two letters denote the number. *ET* is replaced by *e* (singular). In case these four letters are followed by *gr*, it is changed to a *g* denoting an article. This part is optional in both tagsets. The new IFD tag is then constructed by appending the word class *n*, the gender *h*, the number *e*, the case *n* and the optional article *g* and results in *nheng*.

Another example is the entry

lífshættulegur;391052;lo;alm;lífshættuleg;FSB-HK-ÞFFT;

('life-threatening'). The adjective (*lo*) tag consists of three parts. The first part denotes the degree and declension (F->f 'positive', SB->s 'strong declension'), the second part denotes the gender (h 'neuter') and the last part denotes the case and number (ÞF->o 'accusative', FT->f 'plural'). The new tag results in *lhfosf*.

We decided to add only words which do not exist in the IFD but not to add missing tags for those words that were already in the IFD. Adding all available tags would have increased the number of available tags per word form which leads to more

ambiguity. This makes it harder for the tagger to make the right decision. Missing tags are more rare ones. We did indeed test this and it showed that newly introduced errors due to the higher ambiguity for known words outweighed the gain from the new tags for unknown words.

The entries were all without frequency information. For IceTagger, we added new tags in the order in which they appeared. In TnT, which requires frequency information, we added frequency 1 to each tag.

Here is an example from IceTagger's dictionary for *tannkrem* extracted from the IFD:

tannkrem=nheo_nhen
tannkremið=nheng

The following word forms have been found in BÍN and have been added to the dictionary:

tannkrema=nhfe
tannkremanna=nhfeg
tannkremi=nheþ
tannkremin=nhfng_nhfog
tannkreminu=nheþg
tannkrems=nhee
tannkremsins=nheeg
tannkremum=nhfþ
tannkremunum=nhfþg

As can be seen here, the 10 tags for the 9 new word forms have been added but the new tags for the existing word forms have been ignored. Overall, the dictionary was expanded from previously around 55,000 lines to over 2.5 million lines. TnT can handle dictionaries of that size without any problems. For IceTagger we needed to assign more memory to Java in order to use the large dictionary.

The tests were run for all previously used versions of IceTagger and TnT, including the integrated versions (see chapter 2.3). The final results can be seen in table 4.1.

The TnT and IceTagger version names that include BÍN mark the versions including the larger dictionary described above. As one can see (column u.w.r. in table 4.1), the average unknown word ratio goes down to only 1.15%. The remaining unknown words are the most complicated to tag and therefore result in a very low accuracy. Among them are many proper nouns (*Brahms, Gestapó, Frankenstein*), foreign

	u.w.r.	**unkn.**	**known**	**overall**
IceTagger	6.79%	75.30%	92.78%	91.59%
IceTagger BÍN	1.15%	55.93%	92.57%	92.15%
Ice+HMM	6.79%	75.62%	93.01%	91.83%
Ice+HMM BÍN	1.15%	54.82%	92.88%	92.44%
HMM+Ice	6.79%	76.11%	93.39%	92.21%
HMM+Ice BÍN	1.15%	56.24%	93.22%	92.80%
TnT	6.84%	71.82%	91.82%	90.45%
TnT BÍN	1.15%	54.67%	91.57%	91.15%
TnT Filled	6.84%	72.98%	92.60%	91.25%
TnT Filled BÍN	1.15%	54.71%	92.33%	91.89%

Table 4.1: Comparison of the results of all variations of Ice Tagger and TnT.

words (*Aktiengesellschaft, system*), slang and loanwords (*sexý, amatörar, abacus*), abbreviations (*Mhz, nr*) interjections (*ihí-hi, ehem*) and any numeral, in Arabic and Roman numerals or in words, that did not occur during training.

However, this has very little influence on the overall tagging accuracy because of their low ratio. Then again the accuracy for known words decreased as well (compare each two rows of taggers with and without BÍN in table 4.1). This is on the one hand due to the now higher ratio of known words and on the other hand due to tag profiles extracted from BÍN. As mentioned above, we did not have any frequency information for the tags, so the order in which they appear in the profile are random in terms of frequency. In default cases where the most frequent tag is chosen this might lead to a wrong decision. Nevertheless, the overall tagging accuracy for IceTagger was increased by 0.56% and for TnT by 0.70%.

4.2 Designing a New Tagset

As described in chapter 3.2, the Icelandic tagset consists of 700 tags. It was developed during the work on the IFD (Pind et al., 1991). A few changes have been made to the tagset since, mostly reordering of attributes. Hence, the tagset we use for our experiments differs slightly from what is found in the printed version of the IFD.

The aim of this part[3] of the project was to follow up on the work done by Helgadóttir (2005) on simplified tagsets and research which attributes of the tagset could be

[3] This part was a joint work with Hrafn Loftsson, Eirikur Rögnvaldsson and Sigrún Helgadóttir (Loftsson et al., 2009).

skipped without losing important information. The Icelandic tagset is very detailed, as was described in chapter 3.2. Some of the information is repetitive, in other cases, the depth of information that is available may not be needed in many applications. Mapping the Icelandic tagset to a smaller tagset used in related languages is a problem, because the structure of other language tagsets can be very different from ours. Tagsets are specially developed for the needs of a language an can, in most cases, not be inherited from other, even related languages. Therefore, an attempt was made to find dispensable information in the Icelandic tagset and remove it.

As a source, we used the list of frequent errors made by IceTagger from (Loftsson, 2007a). It was extracted from the errors made by an older version of IceTagger than we used in our experiment but it served as a starting point for this experiment. The first few elements of the list are shown in table 4.2. The first column shows the tags proposed by IceTagger and the correct tags from the golden standard as *Proposed tag > Correct tag*.

Proposed tag > Correct tag	Error rate
aþ>ao	2.58%
ao>aþ	1.62%
aa>ao	1.54%
nveþ>nveo	1.41%
aa>aþ	1.32%
ao>aa	0.96%
sfg3fn>sng	0.94%
ct>c	0.93%
nveo>nveþ	0.84%
nhen>nheo	0.79%
nkeþ>nkeo	0.76%

Table 4.2: The most frequent tagging errors made by IceTagger

According to this list, most errors are caused by prepositions that govern accusative (*ao*) and dative (*aþ*) respectively. They are followed by errors occurring due to the confusion of prepositions and adverbs (*aa*). The third biggest source of errors are the nouns (*n...*). As mentioned in chapter 2.1.2.0.1, an Icelandic noun can have several word forms that have the same spelling. An example for each gender is shown in table 4.3.

case	masculine		feminine		neuter	
	sg	pl	sg	pl	sg	pl
nom	lampi	lampar	peysa	peysur	hús	hús
acc	lampa	lampa	peysu	peysur	hús	hús
dat	lampa	lömpum	peysu	peysum	húsi	húsum
gen	lampa	lampa	peysu	peysa	húss	húsa

Table 4.3: Declensions of Icelandic nouns

Most errors that occur from nouns concern distinctions between accusative and dative for masculine and feminine nouns and the distinction between nominative and accusative for neuter nouns. Errors in other cases are less frequent.

Based on table 4.2, we decided what pieces of information from the tagset can be skipped without losing valuable tagging information while gaining tagging accuracy. We used two different ways of testing the reduced tagsets: changing the internal or the external tagset.

The internal tagset denotes the tagset that the tagger uses during tagging. The external tagset, on the other hand, changes the tagset that is used in the output that is returned to the user. Usually, the tagset in the corpus and in the output are the same. With internal mapping the tagset of the corpus is changed and the input text will be tagged containing this new tagset. The output will contain at most these tags. Once the corpus' tagset is mapped, the lost information can not be added again after tagging. However, external mapping does not affect the corpus or the tagging process. It changes the tags of the output text after tagging but before it is returned to the user. That way errors are masked from the user's eyes. With external mapping we do not lose depth of information at tagging but we can adjust the information that is presented to the user and hide errors that occur in details that are unnecessary for the scope of the application.

	unkn.	**known**	**overall**
TnT 700 tags	71.82%	91.82%	90.45%
TnT 452 tags	72.69%	91.83%	90.52%

Table 4.4: Tagging results for TnT for two different internal tagsets

Brants (1997) made an experiment with internal and external mapping and drew the conclusion that external mapping is more effective than internal mapping. Our results are consistent with his conclusions as can be seen in tables 4.4 and 4.5 for TnT. The improvement on the tagging accuracy using internal mapping (using the

same final mapset as described later in this section) was only 0.07% but 1.44% for the same tagset size using external mapping (see bottom row in table 4.5). The changes that were made to the tagset to reach these results will be explained below.

TnT allows internal mapping, whereas the tagset is implemented into IceTagger and therefore changes would have to be made to the program itself[4]. Hence, we can present the difference of internal and external mapping for TnT. For IceTagger we only have the results for external mapping, as presented in Table 4.6.

	TnT	TnT Filled	TnT BÍN	TnT Filled BÍN
unchanged	90.45%	91.25%	91.15%	91.89%
proper nouns	90.52%	91.32%	91.22%	91.97%
ct>c	90.57%	91.37%	91.27%	92.01%
cn,ct>c	90.61%	91.40%	91.31%	92.05%
prep.>af	90.98%	91.76%	91.67%	92.39%
prep.>aa	91.58%	92.31%	92.26%	92.94%
prep wordmap	91.58%	92.31%	92.26%	92.94%
pronouns	90.55%	91.35%	91.24%	91.99%
proper nouns, ct>c, prep.>aa, pronouns	91.90%	92.62%	92.58%	93.25%

Table 4.5: External mapping results for different versions of TnT

[4] A data-driven tagger is trained on the tagset used in the training corpus and thus can be trained on any change of tagset applied to the corpus. A rule-based tagger has previously programmed rules which can not be applied to a tagset change in the lexicon.

	IceTagger	Ice BÍN	Ice +Hmm	Hmm +Ice	Ice +Hmm BÍN	Hmm +Ice BÍN
unchanged	91.59%	92.15%	91.83%	92.21%	92.44%	92.80%
proper nouns	91.66%	92.22%	91.89%	92.28%	92.50%	92.87%
ct>c	91.68%	92.24%	91.92%	92.32%	92.53%	92.90%
cn,ct>c	91.70%	92.26%	91.94%	92.34%	92.55%	92.93%
prep.>af	91.98%	92.53%	92.22%	92.60%	92.82%	93.17%
prep.>aa	92.48%	93.01%	92.74%	93.10%	93.32%	93.66%
prep wordmap	92.48%	93.01%	92.74%	93.10%	93.32%	93.66%
pronouns	91.68%	92.24%	91.93%	92.31%	92.55%	92.90%
proper nouns, ct>c, prep.>aa, pronouns	92.77%	93.30%	93.00%	93.39%	93.59%	93.95%

Table 4.6: External mapping results for different versions of IceTagger

In our experiment, we have tried various kinds of changes to the tagset. Some of them proved more useful than others. The first change that was made was to skip the type of proper noun (see table 4.7). It is very hard for the tagger to make a distinction between personal names, proper nouns and other special nouns. This is usually not done by a tagger but with Named Entity Recognition which is a subarea of Information Extraction. IceTagger does not include this feature, but such a tool has recently been developed (Tryggvason, 2009).

The map file we used was of the following format:

nhee-s nhee–
nhee-ö nhee–
nheegs nheeg-
nheegö nheeg-

The tag in the first column is the tag according to the tagset in the IFD corpus. The tag in the second column is the mapped tag that we are using. In the mapped tag, the last character of the tag (type of proper noun) is replaced by a hyphen. That way there is no distinction between the proper noun types.

The map file is applied after tagging is completed. It is used during evaluation, when the tagged text is compared to the golden standard. Incorrect tags are looked up in the map file and replaced with the new tag if it is found. In the same way a

Char #	Category/Feature	Symbol - signification
1	Word class	n - noun
2	Gender	k - masculine, v - feminine, h - neuter, x - unspecified
3	Number	e - singular, f - plural
4	Case	n - nominative, o - accusative þ - dative, e - genitive
5	Article	g - with suffixed article
6	Proper noun	m - person, ö - place, s - other proper name

Table 4.7: All possible tags of nouns in the Icelandic tagset.

mapfile can be applied after tagging but before returning the output to the user or an application respectively.

Not considering the type of proper nouns, the tagging accuracy went up 0,07% for both taggers (compare row 1 and 2 in tables 4.5 and 4.6).

Another tagset change was to skip the mood of verbs. This did not have any influence at all. Other reductions without effect were strong and weak declension of adjectives and handling percentage numbers as regular numbers. These mappings are not represented in the tables because of their lack of improvement.

Changing a relative conjunction *ct* to a general conjunction *c* proved very useful (see row *ct>c*) whereas in addition changing the sign of infinitive *að* ('to') (*cn*) to a general conjunction did not show a significant improvement (compare rows *cn,ct>c* and *unchanged*). The reason for this is that there are only two words in Icelandic which can have the tag *ct*: *sem* and *er*. *Er* can either be a verb or a relative conjunction and is therefore easy to distinguish from its position in the sentence. *Sem* on the other hand can be either a relative conjunction or a comparative conjunction, which appear in similar environments in a sentence and are hard to distinguish even for linguists. *Að* is either the sign of infinitive, a general conjunction, a preposition or an adverb. Distinguishing the first two cases is not a problem because they do not appear in the same position in a sentence. The problem distinguishing the latter two cases will be discussed here below.

The most frequent tagging errors that occurred involve prepositions and adverbs. Some words can be prepositions or adverbs respectively, which makes it harder for the tagger to decide which one it is. In some actual applications, missing this distinction may not lead to any problems, mostly because the meaning of the word is determined by the context and not by the word form. These words have multiple meanings

which overlap between the two word forms. Very often they are also part of phrases. Now the challenge was to find the best way to skip this distinction without losing precision.

One approach was to tag all prepositions with a new tag *af* which was not part of the original tagset and which does not contain case information. This led to a 0.53% increase in overall tagging accuracy for TnT and 0.39% for IceTagger (compare rows *prep.>af* and *unchanged*).

Another approach was to extract a list of prepositions which also occur as adverbs and use a wordmap file[5] to change their tags to *af*. Unlike the previous approach we will not be able to see the difference between an adverb and a preposition in this one. The wordmap file is of this structure:

á	a[aoþe][em]? af
án	a[aoþe][em]? af
ásamt	a[aoþe][em]? af
í	a[aoþe][em]? af

The first column shows the word that is affected, the second column contains a regular expression representing all possible adverbial or prepositional tags and the third column contains the new tag that will replace the old one. In our case, that is *af* for each of these words. This led to an overall increase in accuracy of 1.13% for TnT and 0.89% for IceTagger (compare rows *prep wordmap* and *unchanged*).

In the light of these improvements we made an approach to tag all prepositions as adverbs (*aa*) without case information, using a regular map file. This gave the same results as the wordmap file (compare rows *prep.>aa* and *prep wordmap*). The main reason is, that errors between adverb and preposition tags occur only for words which can be prepositions as well as adverbs, but not for words which are either prepositions or adverbs. The wordmap approach assigns the same tag to exactly those critical cases whereas the mapping approach unifies two word types to one. This makes no difference to the tagger, for it uses the same tag for all the above mentioned words, regardless what the tag is called.

Since the results were equally good and using mapfiles only makes implementation easier than mixing mappings and word maps, the latter approach was decided upon to be the most applicable.

[5] Unlike map files, where one tag is exchanged by another tag in the whole document, a tag in a wordmap file is exchanged with a different tag only in the context of a certain word.

Pronouns are divided into seven types. This information is unnecessary in most cases because most pronouns are not ambiguous among types. There are only a few which appear in more than one type, e.g. *mín* can either be a personal pronoun or a possessive pronoun. To make a distinction is rather hard for a PoS tagger because in most cases this needs analysis of the context rather than syntactic analysis. Ignoring the type of pronouns resulted in a 0.1% increase of accuracy for TnT and 0.09% for IceTagger (compare rows *pronouns* and *unchanged*).

After evaluating all approaches, the final set of reductions was decided to be as follows:

- No distinction between types of proper nouns.
- No distinction between types of pronouns.
- Relative conjunctions *ct* are tagged as general conjunctions *c*.
- No distinction between prepositions and adverbs (*aa*).

We combined those four changes into one map file. The overall tagging accuracy for IceTagger increased by 1.18% and by 1.44% for TnT (compare the first and the last rows in tables 4.5 and 4.6). The changes we made resulted in a signifficant change in the size of the tagset. The number of possible tags for nouns was reduced from 224 to only 48 tags, the tags for pronouns were reduced from 184 to 40, the prepositions saved another 7 tags and the conjunctions 1 tag. This results in a new, smaller tagset of only 452 tags. Table 4.8 shows the detailed results for each of the four map files as well as the combination of all of them. The best results are reached by the integrated version HMM+Ice of IceTagger, using BÍN, (93.95%) and the integrated tagger TnT Filled, also using BÍN (93.25%).

As one can see, IceTagger increases its accuracy about 1.2% whereas TnT's increases approximately by 1.4%. TnT benefit slightly more than IceTagger, as it also produces more tagging errors to begin with. When we look closer at the detailed numbers in table 4.8 (compare rows 1 and 2 of each tagger) we can see that the accuracy for unknown words went up between 0.91% to 5.35% when ignoring the type of proper nouns, whereas the tagging accuracy for known words remained almost the same. This is due to the fact that the type of unknown proper nouns can not be analyzed by a PoS tagger. Therefore IceTagger assigns all unknown proper nouns the same type which results in alot of errors. Known proper nouns on the other hand only result in errors if the same name can be for example a place and a personal name (i.e. the female name Katla is also the name of a volcano).

	unkn.	known	overall	unkn.	known	overall
	IceTagger			Ice+HMM		
unchanged	75.30%	92.78%	91.59%	75.62%	93.01%	91.83%
proper nouns	76.21%	92.78%	91.66%	76.37%	93.02%	91.89%
ct>c	75.30%	92.88%	91.68%	75.62%	93.11%	91.92%
prep.>aa	75.30%	93.73%	92.48%	75.62%	93.98%	92.74%
pronouns	75.30%	92.88%	91.68%	75.62%	93.12%	91.93%
all	76.21%	93.97%	92.77%	76.37%	94.21%	93.00%
	IceTagger BÍN			Ice+HMM BÍN		
unchanged	55.93%	92.57%	92.15%	54.82%	92.88%	92.44%
proper nouns	60.75%	92.58%	92.22%	58.76%	92.90%	92.50%
ct>c	55.93%	93.44%	93.01%	54.82%	93.76%	93.32%
prep.>aa	55.93%	92.89%	92.46%	54.82%	93.21%	92.77%
pronouns	55.93%	92.66%	92.24%	54.82%	92.98%	92.55%
all	60.75%	93.68%	93.30%	58.76%	93.99%	93.59%
	HMM+Ice			HMM+Ice BÍN		
unchanged	76.11%	93.39%	92.21%	56.24%	93.22%	92.80%
proper nouns	77.04%	93.39%	92.28%	61.14%	93.24%	92.87%
ct>c	76.11%	93.50%	92.32%	56.24%	93.33%	92.90%
prep.>aa	76.11%	94.34%	93.10%	56.24%	94.09%	93.66%
pronouns	76.11%	93.49%	92.31%	56.24%	93.32%	92.90%
all	77.04%	94.59%	93.39%	61.14%	94.33%	93.95%
	TnT			TnT Filled		
unchanged	71.82%	91.82%	90.45%	72.98%	92.60%	91.25%
proper nouns	72.74%	91.83%	90.52%	73.89%	92.60%	91.32%
ct>c	71.82%	91.95%	90.57%	72.98%	92.72%	91.37%
prep.>aa	71.83%	93.03%	91.58%	72.99%	93.73%	92.31%
pronouns	71.83%	91.92%	90.55%	72.98%	92.70%	91.35%
all	72.75%	93.31%	91.90%	73.90%	94.00%	92.62%
	TnT BÍN			TnT BÍN Filled		
unchanged	54.67%	91.57%	91.15%	54.71%	92.33%	91.89%
proper nouns	60.02%	91.59%	91.22%	59.92%	92.34%	91.97%
ct>c	54.67%	91.70%	91.27%	54.71%	92.45%	92.01%
prep.>aa	54.72%	92.69%	92.26%	54.75%	93.39%	92.94%
pronouns	54.67%	91.67%	91.24%	54.71%	92.42%	91.99%
all	60.07%	92.95%	92.58%	59.97%	93.64%	93.25%

Table 4.8: This table shows the detailed tagging results for external mapping tested on all versions of TnT and IceTagger that were used in our experiment

Proposed tag > Correct tag	Error rate
nveþ>nveo	1.79%
nveo>nveþ	1.12%
sfg3fn>sng	1.09%
nhen>nheo	0.98%
nkeþ>nkeo	0.97%
nheo>nhen	0.81%
c>aa	0.78%
nkeo>nkeþ	0.77%
aa>lhensf	0.77%
sng>sfg3fn	0.73%
foheo>fohen	0.70%

Table 4.9: The most frequent tagging errors made by IceTagger after using external mapping with the final mapset.

The opposite observation can be made on the other three mappings. They do not change the tagging accuracy for unknown words (except 0.01% for TnT for pronouns) but have a rather big influence on the known word accuracy. These three mappings affect solely closed word classes. A closed word class is a word class where no new words can be added. These are prepositions, conjunctions or pronouns. In contrast, words can always be added to open word classes such as nouns, adjectives or verbs. Members of closed classes are not only limited in number, most of them are also quite frequently used. So they will be represented close to completeness in a sufficiently large corpus and therefore are no candidates for unknown words.

Some of the most frequent tagging errors shown in table 4.2 should have vanished after using the above described external mapping. For comparison, table 4.9 shows the new distribution of errors. All errors between two prepositions or prepositions and adverbs disappeared from the list, as well as the conjunction error. Some of the most common errors that are left, are wrong cases for nouns.

4.3 Improving the Accuracy of IceTagger

One of the main sources of tagging errors are the unknown words. Many of which are not really unknown but the word form in which they appear has not appeared during training. One Icelandic verb can have diverse appearances as the example of the verb ”vera” ’to be’ shows (see table 4.10).

Indicative	
Present Tense	**Past Tense**
er erum	var vorum
ert eruð	varst voruð
er eru	var voru
Subjunctive	
Present Tense	**Past Tense**
sé séum	væri værum
sért séuð	værir væruð
sé séu	væri væru
Imperative	**Present Participle**
ver	verandi
vertu	**Supine**
verið	verið

Table 4.10: Conjugation table for the verb *vera* (to be)

As mentioned in chapter 3.1.2, Ingason et al. (2008) have developed a lemmatizer which returns the corresponding lemma for the word form. We tried to make use of this lemmatizer for unknown verb forms.

Many Icelandic verbs govern the case of their object. Some of them can even govern several cases depending on the context. IceTagger uses a so called verbObject list which contains a list of verbs paired up with the most frequent case of their objects. This list is extracted from the IFD during training. We were trying to extend this list and change the structure of the list in order to get more hits during lookup.

First, we prepared a list of lemmas for all known verbs which appear in a SVO or OVS[6] structure in the training corpus. For this purpose, we used the output of IceParser which provides us with the syntactic information of the sentences. As shown in the example from chapter 3.1.1

*{*SUBJ> [NP vagnstjórinn NP]*SUBJ>} [VP sá VP]*
*{*OBJ< [NP mig NP] *OBJ<}*
('The driver saw me')

the arrows point from the subject and object respectively towards the verb. With a search pattern looking for exactly these structures in a sentence

*SUBJ> [VP (verb) VP] *OBJ< (case of object)

[6] The word order in Icelandic is free as long as the verb stays in second position. The subject and object can change place without changing the meaning of the sentence.

afbaka=o
afgirt=o
afgreiddi=o
afgreiddir=o
afgreiddu=o
afgreitt=o
afgreiða=o
afhausa=o

Table 4.11: An extract from the old verbObject list used by IceTagger

(case of object) *OBJ> [VP (verb) VP] *SUBJ<

we extracted a list of verb and case pairs. We used the following pattern for the two different word orders:

```
          SVO                                    OVS
if (string contains "SUBJ>}")            if(string contains "{*OBJ>")
 if (string contains "{*OBJ<")            if (string starts with "[NP(.)")
  if (string starts with "[NP.")           if (string contains "{*SUBJ<")
   if (string contains "VP[^bgp]?]")        if (string contains "VP[^bgp]?]")
```

```
. stands for any character
[^bgp]?] none or any character but b, g or p
```

The verb matching pattern in the last row is choosing all verb phrases which do not contain a present or past participle or a verb which demands a predicate nominative.

The verbObject list used by IceTagger contains verbs in the form they were found during training with the most common object case they occurred with (see table 4.11). When a word form of one of these verbs is missing, the object case can not be found in the list even though the information is present. We therefore changed the format of the list to pairs of infinitive[7] and case which results in only one line per verb. We used Lemmald to lemmatize all verbs from the verbObject list and the verbs we extracted with the above described algorithm. The verb-case pairs were then counted and for each verb, the verb-case pair that had the highest count was chosen. Both lists of verbs were then joined to a new verbObject list. An extract from the new list is shown in table 4.12.

[7] The infinitive is equivalent to the lemma of a verb.

afbaka=o
afgirða=o
afgreiða=o
afhausa=o
afhenda=o
afhjúpa=o

Table 4.12: An extract from the newly generated verbObject list

One problem that occurres in this list is that many verbs have a subcategorization frame[8] containing a direct and an indirect objects, but our verbObject list only contains one object case per verb. Furthermore, the word order in Icelandic is quite free so the two objects do not necessarily appear in the same order. The cases in which the objects appear in, define the context and not the order in which they appear in a sentence. Two examples from the IFD are shown here:

*{*SUBJ> [NPn ég fp1en NP] *SUBJ>} [VP gleymdi sfg1eþ VP] [VPi að cn segja sng VPi] {*IOBJ< [NPd þér fp2eþ NP] *IOBJ<} {*OBJ< [NPa það fpheo NP] *OBJ<} [PP í ao [NPa morgun nkeo NP] PP] . .*
'I forgot to tell you(*IOBJ*) this(*OBJ*) in the morning'

*{*SUBJ> [NPn fólk nhen NP] *SUBJ>} [VP glottir sfg3en VP] [CP og c CP] [VP pukrar sfg3en VP] [PP með ao [NPa orðin nhfog NP] PP] [SCP þegar c SCP] {*SUBJ> [NPn það fphen NP] *SUBJ>} [VP segir sfg3en VP] [SCP að c SCP] [AdvP kannski aa AdvP] [VP hafi svg3en VP] *SUBJ< [NPn kerlingin nveng NP] *SUBJ< [VPs selt ssg VPs] {*OBJ< [NPa sál nveo sína feveo NP] *OBJ<} {*IOBJ< [NPd andskotanum nkeþg NP] *IOBJ<} . .*
'People (... stare and whisper ...) when they say that maybe the woman has sold her soul (*OBJ*) to the devil (*IOBJ*)'

The indirect object (*IOBJ*) is an optional second object. It can not replace the main object *OBJ* but it can accompany it. As can be seen in the two examples, in some cases the *OBJ* comes first and in others the *IOBJ*. Therefore, the cases in the verbObject list often refer to the second object and are producing new errors when applied to the first object during tagging. This can result in wrong noun cases which are upon the most common tagging errors (see table 4.9, rows with noun (n....) tag errors).

[8] A subcategorization frame of a word contains the number and type of its arguments. E.g. in the sentence *Vagnstjórinn sá mig* the verb *sá* 'saw' has the subcategorization frame *nom. Subj.* v *acc. Obj.*

Another problem that occurres is that sometimes a verb did not have an object, but in another part of the sentence an object, not associated with this verb, was found. These cases also resulted in errors in the list.

The third source of errors is the heuristic that guesses the object case in IceTagger (see section 2.1.2.0.3). Many objects are marked incorrectly by this heuristic. Loftsson (2006a) states that as many as 25% of all objects are marked incorrectly. This leads to errors where the object case is forced on a phrase which is not even the object of the verb. Errors arising from this also result in noun case errors, which can be found in table 4.9. Improving IceTagger in a way, that it could mark objects with higher accuracy, would avoid alot of those errors. If we, in addition, had a handcrafted verbObject list we would improve the tagging results even more.

In order to be able to use this new designed list we had to make changes to the source code of IceTagger. In the function where the verb is usually looked up in the verbObject list, we had to add a call to Lemmald which would lemmatize the verb first and return its infinitive form. The infinitive is then looked up in the list and the case information is returned. With our automatically extracted list, the results for these tests were rather poor. The tagging accuracy stayed more or less the same. We got no noteworthy results from this approach.

Furthermore, I tried to use Lemmald from within IceTagger to look up verbs that did not occur in our list. Verbs are one of the open word classes. They can be created by joining a verb and a prefix or a verb with another word, e.g. an adjective. The idea was to cut off letters in the beginning of the verb, one by one, send the resulting word into Lemmald and if it returns a lemma, look it up in the verbObject list. This is repeated until either a verb is found in the verbObject list or a minimum number of remaining characters is reached.

As an example we can look at the verb *rita* ('to write') which can be compound with a prefix, e.g. the preposition *undir* ('under') which results in the verb *undirrita* ('to sign'). There is a limited amount of prepositions and other prefixes a verb can be appended to, but, as mentioned above, a verb can also be compound with other word types. New creations are allowed, therefore we will never have a complete list of verbs or even of possible compound candidates. An example is the verb *ljósrita* ('to copy') which is a compound word of the noun *ljós* ('light') and the verb *rita*. The method I applied here was to cut off letters in the beginning of the word and sent the resulting word to Lemmald. It takes the following steps:

The first three tries did not result in Icelandic words but the forth is the verb *rita* which is also part of the verbObject list. I then take the object case, associated with *rita* and apply it to *ljósrita*.

ljósrita	unkn.
jósrita	unkn.
ósrita	unkn.
srita	unkn.
rita	rita

All three verbs from the example above, rita, ljósrita and undirrita, take an accusative object. It is not always the case that the compound verb takes the same object case as the verb itself. Nevertheless, this method returns the correct case for many compound verbs. Unfortunately this also did not show the results I hoped for. It affected far too few words to see a change in the tagging accuracy.

4.4 Combination of Taggers

There are many different taggers that can be trained for tagging Icelandic. Most of their results are not very impressive, their accuracy is usually around 90% (see table 3.4), but each of them tags something right that the others tag wrong. Therefore one idea was to combine different taggers and try which combination of taggers gives the best results.

In this experiment, we used combinations of IceTagger, TnT, fnTBL, MBT, MXPOST, TreeTagger and the BI+WC+CT[9]. For more details on the functionality of these taggers see chapter 2. Using the biggest number of available taggers does not necessarily give the best results even though the overall tagging accuracy increased from 92.80% (Loftsson, 2006b) to 92.95% (rows 1-5 of table 4.13).

I used a simple voting scheme for evaluation. All taggers get one vote, where each vote has the same weight. In many cases, this will lead to a clear result but in case of a voting tie, I used a weighted voting scheme between three taggers. The highest tagger gets 3 extra points, the second highest gets 2 extra points and the third one gets 1 extra point in case their tag matches one of the tags that ended up in the tie. The points are added to the previous votes. In case this also does not return a clear

[9] BI+WC+CT will be referred to as BI in all tables.

result, IceTaggers tag is chosen as default. The order of the taggers in the tables below represents the weight each tagger had, i.e. the first one named is the highest rated, the second tagger, the second highest and so forth.

The three taggers used for weighted voting are not fixed. The original idea was to choose the three taggers with the highest accuracy, which would be

BI+WC+CT	-	*IceTagger*	-	*TnT*
(92.06%)		*(91.59%)*		*(90.45%)*

In order to be sure that there is no better combination available, I rotated the best four taggers BI+WC+CT, IceTagger, TnT and fnTBL (89.33% (Loftsson, 2006b)) and combined them with a different number of taggers with lower accuracy (TreeTagger: 89.30% MBT: 89.28,% MXPOST: 89.03% (Henrich et al., 2009)) .

It turned out that even though TreeTagger and MBT have slightly higher tagging results than MXPOST as separate taggers, in combination they decrease the overall tagging accuracy. Helgadóttir (2005) discovered that MXPOST leads better results when distinguishing between identical word forms (e.g. accusative, dative and genitive singular for weak masculine and feminine nouns (see 4.3)) than TnT. On the other hand can taggers, even if they give better overall results than others, cause a wrong tag to win the voting process if they make tagging errors in overlapping cases. When leaving either MBT or TreeTagger (rows 6-8) or both (row 11) of the taggers, out of the voting, the accuracy increases considerably to up to 93.38%.

The combinations with BI+WC+CT produced an unexpected result. As a single tagger it attained the highest tagging accuracy (92.06%) but when provided with the highest vote, the results stay slightly behind the voting scheme $IceTagger-TnT-BI$ (see rows 11 and 13). Also when chosen as a default for undecided tags it reduces the overall tagging accuracy compared to when using IceTagger as default (compare rows 11/12 and 13/14). The best result was reached when putting BI+WC+CT in second place between IceTagger and TnT and using IceTagger as default. This gave an overall accuracy of 93.40% (see row 15).

The results have been compared to CombiTagger, which was developed by Henrich et al. (2009). In most cases the results were comparable. Only when BI+WC+CT was chosen as the highest, the results differed significantly. As mentioned before, the accuracy dropped when the tag of BI+WC+CT was chosen as default for undecided tags. CombiTagger chooses the highest rated tagger as default, whereas in my voting

program I can pick a default tagger independently of the voting sequence (compare rows 13 and 14).

Although BI+WC+CT reaches high accuracy overall, the tagging results for unknown words are better in IceTagger. I tested the effect of a simple rule where in undecided cases IceTagger is chosen for unknown words and BI+WC+CT is chosen for known words. This did not have the expected effect. It even lowered the tagging accuracy slightly for known words as can be seen in rows 16 and 17.

The last two rows in table 4.13 show tagging results for a combination of integrated taggers. At this point, I only tested the scheme $Ice-BI-TnT-fnTBL-MXPOST$ with IceTagger as the default tagger because it proved to be the best combination of taggers. The highest tagging accuracy reached was 93.76% with using Ice+HMM (see row 18). The accuracy fell by 0.02% when using HMM+Ice instead (see row 19), even though as a single tagger it has the slightly higher accuracy.

The highest tagging results by far, or 94.14%, were reached by the combination HMM+Ice BÍN + BI + TnT Filled BÍN + fnTBL* + MX (see table 4.14). Here, I not only used the integrated taggers HMM+Ice, TnT Filled and fnTBL* but also the bigger dictionary BÍN. Unlike the results without BÍN, here the combination with HMM+Ice resulted in a slightly higher accuracy than the combination with Ice+HMM.

The words marked as unknown differ a little among the taggers. I used the unknown words from IceTagger as a reference. The unknown word ratio is not shown in table 4.13 because it is constant at 6.79% at average (equivalent to the numbers shown in table 3.2) and does not vary upon the different combinations. The results for combinations using BÍN are therefore presented in a separate table (see table 4.14) where the accuracy of unknown words is calculated based on the 1.15% unknown word ratio from IceTagger with BÍN even though BI, fnTBL and MXPOST have a much higher unknown word ratio.

As mentioned in chapter 3.4.2.4, CombiTagger calculates the number of words which have not been given a correct tag by any of the taggers. For six taggers this was 2.29% of the words (Henrich et al., 2009). In my experiment I used one more tagger, the bidirectional tagger, which reduced this number to 2.04%. However, this still does not allow a higher tagging accuracy than 97.96%.

In a final step, I applied the external mapping from chapter 4.2 to the combinations that showed the best results. The overall tagging accuracy reached for Ice+HMM + BI + TnT Filled + fnTBL* + MX (combination in row 18 of table

		unkn.	known	overall
1	Ice + TnT + fnTBL + MBT + MX	76.74%	93.97%	92.80%
2	Ice + TnT + fnTBL + MBT + MX + Tree	77.03%	93.86%	92.72%
3	Ice + TnT + fnTBL + BI + MBT + MX + Tree	77.20%	94.10%	92.95%
4	Ice + TnT + BI + fnTBL + MBT + MX + Tree	77.22%	94.10%	92.95%
5	BI + Ice + TnT + fnTBL + MBT + MX + Tree	76.93%	94.12%	92.95%
6	Ice + TnT + BI + fnTBL + MBT + Tree	77.85%	94.09%	92.98%
7	Ice + TnT + BI + fnTBL + MX + Tree	78.16%	94.30%	93.21%
8	Ice + TnT + BI + fnTBL + MBT + MX	77.78%	94.40%	93.27%
9	Ice + TnT + BI + fnTBL + MBT	77.08%	93.88%	92.74%
10	Ice + TnT + BI + fnTBL + Tree	78.09%	94.04%	92.96%
11	Ice + TnT + BI + fnTBL + MX	78.20%	94.49%	93.38%
12	Ice + TnT + BI + fnTBL + MX. BI default	78.11%	94.48%	93.37%
13	BI + Ice + TnT + fnTBL + MX	77.50%	94.49%	93.33%
14	BI + Ice + TnT + fnTBL + MX. BI default	76.54%	94.48%	93.26%
15	Ice + BI + TnT + fnTBL + MX	78.08%	94.52%	93.40%
16	Ice + BI + TnT + fnTBL + MX + Rule	78.08%	94.51%	93.40%
17	BI + Ice + TnT + fnTBL + MX + Rule	77.50%	94.48%	93.33%
18	Ice+HMM + BI + TnT Filled + fnTBL* + MX	79.76%	94.78%	93.76%
19	HMM+Ice + BI + TnT Filled + fnTBL* + MX	79.80%	94.76%	93.74%

Table 4.13: Results from various combination of the 7 taggers

	unkn.	known	overall
Ice BÍN + BI + TnT BÍN + fnTBL + MX	64.88%	94.35%	93.80%
Ice+HMM BÍN + BI + TnT Filled BÍN + fnTBL* + MX	61.17%	94.50%	94.12%
HMM+Ice BÍN + BI + TnT Filled BÍN + fnTBL* + MX	61.96%	94.51%	94.14%

Table 4.14: Results from various combination of the 7 taggers including the BÍN

4.13) with mapping was 94.62% and 94.99% for the version using HMM+Ice and BÍN (combination in the last row of table 4.14).

Chapter 5

Conclusions

The purpose of this project was to increase the tagging accuracy for Icelandic. It consisted of four main parts. The first part, was to reduce the tagset without losing valuable information. Several reductions were tried and a final set has been decided upon:

- No distinction between types of proper nouns.
- No distinction between types of pronouns.
- Relative conjunctions are tagged as general conjunctions.
- No distinction between prepositions and adverbs.

These changes resulted in a reduction of the tagset from 700 tags to 452 tags and an increase in tagging accuracy of 1.18% for IceTagger (92.77%) and 1.44% for TnT (91.89%).

The results indicate that external mapping can be a useful tool in different kinds of applications. It will be a future task to develop different map sets, specially designed for the needs of certain applications to increase the underlying tagging accuracy.

The second part of the project was to test the effects of a bigger dictionary on the tagging accuracy. Therefore we extracted all personal names and regular nouns, adjectives, adverbs, indicative and subjunctive verbs, infinitive, present participle and supine verbs in active or middle voice and past participle active verbs in nominative from BÍN and added them to the dictionaries that IceTagger and TnT extracted from the IFD during training. IceTagger increased its tagging accuracy to 92.15%

and TnT to 91.15%. The highest accuracies were reached by using BÍN with the integrated versions HMM+Ice (92.80%) and TnT Filled (91.89%).

This experiment showed the potential that lies in using a bigger corpus. In our case, we only had a dictionary without any frequency information. If we would have a corpus containing those words, we would get the necessary frequency information that the taggers would need to increase the tagging accuracy even more over the same size of dictionary.

The external mapping from part one was also evaluated on the taggers using BÍN. It resulted in a tagging accuracy of 93.30% for IceTagger and 92.56% for TnT. The highest accuracies were reached by the integrated taggers HMM+Ice (93.95%) and TnT Filled (93.24%) adding BÍN.

The third part, using the lemmatizer, did not result in an improved tagging accuracy. One of the reasons is the fixed size subcategorization frame that is used. IceTagger uses a verbObject list that only takes one object for each verb into account. A more flexible frame size could help avoid the mix ups between the different objects. This would require changes in the source code of IceTagger in order to be able to use a verbObject list containing more case informations. A second approach to increase the effects of this part would be a hand crafted verbObject list to minimize the errors introduced by the automatically created list.

In part four, I tried various combinations of taggers. It turned out that giving the best tagger the highest weight does not necessarily give the best results. On the contrary, giving IceTagger, the second best tagger, the highest weight and BI+WC+CT, the best tagger, the second highest weight, turned out to be the best combination. Moreover, adding more and more taggers to a voting scheme does not necessarily increase the tagging accuracy. Too many taggers can add up in too many wrong votes and resulting in a decrease of accuracy. Therefore, we left out TreeTagger and the MBT tagger and used only 5 taggers: IceTagger, BI+WC+CT, TnT, fnTBL and MXPOST. This combination resulted in an accuracy of 93.40%. The combination of integrated taggers resulted in 93.76% and adding BÍN to the integrated taggers gave 94.14% overall tagging accuracy. When applying the external mapping from part one to the best combinations, the accuracy reached 94.62% for the combination of integrated taggers Ice+HMM + BI + TnT Filled + fnTBL* + MX and 94.99% for the version HMM+Ice BÍN + BI + TnT Filled BÍN + fnTBL* + MX.

I'd suspect that it would be possible to increase this accuracy even more with the development of rules, naming a preferred tagger for certain decisions, such as

case decisions for nouns or person decisions for verbs. Nevertheless, this method is restricted by a maximum 97.96% accuracy, induced by the 2.04% of words tagged incorrectly by all taggers.

To increase this number we would need to add a better tagger for Icelandic than we have right now. A way to achieve that could be an integration of IceMorphy into BI+WC+CT which has the highest tagging accuracy for known words of a single tagger (93.70%) but has a low accuracy for unknown words compared to the other taggers. This integration might increase the tagging accuracy for unknown words and therefore increase the overall tagging accuracy.

Another way to increase tagging accuracy using a combination of taggers could be the automatic annotation of a large corpus. The accuracy of 94.14% reached by the best combination is not perfect but it would still provide us with information to calculate frequencies from. Although this might introduce new errors, especially for word classes with case information, it could prove useful for less frequent words or for previously unknown words.

Another possible solution is to follow the idea of Constraint Grammar to combine tagging and parsing into one step. If a part of a sentence is unclear during tagging, it could be analyzed by a parser and sent back to the tagger with the additional information. This would require changes to the source code of IceTagger and IceParser and therefore goes beyond the scope of this project.

In any case, the results of our methods bring us much closer to the accuracies reached for other languages. That gives me the reason to believe that a bigger dictionary, a smaller tagset, and a well coordinated combination of tagging methods can increase the tagging accuracy for Icelandic up to a level where it becomes comparable to other languages.

Bibliography

Bjarnadóttir, K. (16. November 2007). *Um Beygingarlýsingu íslensks nútímamáls.* `http://bin.arnastofnun.is/um_BIN.php`.

Bjarnadóttir, K. (2004). Beygingarlýsing íslensks nútímamáls. In *Samspil tungu og tækni* (pp. 23–25). Menntamálaráðuneytið.

Bjarnadóttir, K. (2005). Modern Icelandic Inflections. In H. Holmboe (Ed.), *Nordisk sprogteknologi 2005* (pp. 49–50). Copenhagen: Museum Tusculanums Forlag.

Brants, T. (1997). Internal and External Tagsets in Part-of-Speech Tagging. In *Proc. Eurospeech '97* (pp. 2787–2790). Rhodes, Greece.

Brants, T. (2000). TnT – A statistical Part-of-Speech tagger. In *Proceedings of the* 6^{th} *applied nlp conference.* Seattle, WA.

Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. *Computational Linguistics*, *21*(2), 543–565.

Chanod, J.-P., & Tapanainen, P. (1994). *Statistical and constraint-based taggers for french* (Tech. Rep.). Research Paper, Rank Xerox Research.

Daelemans, W., Zavrel, J., Berck, P., & Gillis, S. (1996). MBT: A Memory-Based Part of Speech Tagger-Generator. In *Proc. of Fourth Workshop on Very Large Corpora* (pp. 14–27). ACL SIGDAT.

Dredze, M., & Wallenberg, J. (2008). *Further Results and Analysis of Icelandic Part of Speech Tagging.* Department of Computer and Information Science, University of Pennsylvania.

Hagen, K., Johannessen, J. B., & Nøklestad, A. (2000). A Constraint-Based Tagger for Norwegian. In C.-E. Lindberg & S.-N. Lund (Eds.), 17^{th} *Scandinavian Conference of Linguistics. Odense Working Papers in Language and Communication* (Vol. 19, pp. 31–48). Odense: University of Southern Denmark.

Halteren, H. v., Daelemans, W., & Zavrel, J. (2001). Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, *27*(2), 199–230.

Helgadóttir, S. (2005). Testing Data-Driven Learning Algorithms for PoS Tagging of Icelandic. In H. Holmboe (Ed.), *Nordisk sprogteknologi 2004* (pp. 257–265). Copenhagen: Museum Tusculanums Forlag.

Henrich, V., Reuter, T., & Loftsson, H. (2009). CombiTagger: A System for Developing Combined Taggers. In *Proceedings of the* 22^{nd} *International FLAIRS Conference, Special Track: Applied Natural Language Processing.* Sanibel Island, Florida, USA.

Ingason, A. K., Helgadóttir, S., Loftsson, H., & Rögnvaldsson, E. (2008). A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI). In *GoTAL '08: Proceedings of the* 6^{th} *international conference on Advances in Natural Language Processing* (pp. 205–216). Berlin, Heidelberg: Springer-Verlag.

Ingason, A. K., Jóhannsson, S. B., Helgadóttir, S., Loftsson, H., & Rögnvaldsson, E. (2009). Context-Sensitive Spelling Correction and Rich Morphology. In *Proceedings of the* 17^{th} *Nordic Conference of Computational Linguistics (NoDaLiDa-2009)* (pp. 231–234). Odense, Denmark.

Karlsson, F., Voutilainen, A., Heikkilä, J., & Anttila, A. (1995). *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text.* Berlin: Mouton de Gruyter.

Krauwer, S. (2003). The Basic Language Resource Kit (BLARK) as the first milestone for the language resources roadmap. In *Specom' 2003 international workshop speech and computer.* Moscow, Russia.

Krauwer, S. (May 1998). *ELSNET and ELRA: common past, common future.* ELRA Newsl 3.2. (`http://www.elsnet.org/dox/blark.html`. Accessed April 18, 2009.)

Loftsson, H. (2006a). Tagging a Morphologically Complex Language Using Heuristics. In T. Salakoski, F. Ginter, S. Pyysalo, & T. Pahikkala (Eds.), *Advances in Natural Language Processing,* 5^{th} *International Conference on NLP, FinTAL 2006, proceedings.* Turku, Finland.

Loftsson, H. (2006b). Tagging Icelandic text: an experiment with integrations and combinations of taggers. *Language Resources and Evaluation*, *40*(2), 175–181.

Loftsson, H. (2007a). *Tagging and Parsing Icelandic Text*. Unpublished doctoral dissertation, University of Sheffield, Sheffield, UK.

Loftsson, H. (2007b). Tagging Icelandic Text using a Linguistic and a Statistical Tagger. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics.* Rochester, NY, USA.

Loftsson, H. (2008a). IceNLP: A Natural Language Processing Toolkit for Icelandic, User Guide [Computer software manual]. Reykjavík, Iceland. (`http://www.ru.is/faculty/hrafn/Papers/IceNLP_manual.pdf`)

Loftsson, H. (2008b). Tagging Icelandic text: A linguistic rule-based approach. *Nordic Journal of Linguistics*, *31*(1), 47–72.

Loftsson, H., Kramarczyk, I., Helgadóttir, S., & Rögnvaldsson, E. (2009). Improving the PoS tagging accuracy of Icelandic text. In *Proceedings of the* 17^{th} *Nordic Conference of Computational Linguistics (NoDaLiDa-2009)* (pp. 103–110). Odense, Denmark.

Loftsson, H., & Rögnvaldsson, E. (2007a). IceNLP: A Natural Language Processing Toolkit for Icelandic. In *Proceedings of InterSpeech 2007, Special Session: Speech and language technology for less-resourced languages.* Antwerp, Belgium.

Loftsson, H., & Rögnvaldsson, E. (2007b). IceParser: An Incremental Finite-State Parser for Icelandic. In *Proceedings of the* 16^{th} *Nordic Conference of Computational Linguistics (NoDaLiDa 2007).* Tartu, Estonia.

Manning, C. D., & Schütze, H. (2002). *Foundations of Statistical Natural Language Processing*. The MIT Press. (chapter 9-10)

Marcus, M., Santorini, B., & Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, *19*(2), 313–330.

Megyesi, B. (2002). *Data-driven Syntactic Analysis: Methods and Applications for Swedish*. Unpublished doctoral dissertation, KTH, Stockholm, Sweden.

Ngai, G., & Florian, R. (2001). Transformation-Based Learning in the Fast Lane. In *Proceedings of the* 2^{nd} *conference of the north american chapter of the acl.* Pittsburgh, PA, USA.

Ólafsson, R., Rögnvaldsson, E., & Sigurðsson, . (1999). *Tungutækni. Skýrsla starfshóps [Language Technology. Report of a Committee]* (Tech. Rep.). Reykjavík, Iceland: Ministry of Education, Science and Culture.

Pind, J., Magnússon, F., & Briem, S. (1991). *Íslensk orðtíðnibók [The Icelandic Frequency Dictionary].* Reykjavik: The Institute of Lexicography, University of Iceland.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*, 257–286.

Ratnaparkhi, A. (1996). A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* Philadelphia, PA, USA.

Rögnvaldsson, E. (2008). Icelandic Language Technology Ten Years Later. In *Collaboration: Interoperability between people in the creation of language resources for less-resourced languages* (pp. 1–5). Marrakech, Marocco.

Russell, S. J., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach.* Prentice-Hall.

Schmid, H. (1994). Probabilistic Part–of–Speech Tagging Using Decision Trees. In *Proceedings of the Conference on New Methods in Language Processing.* Manchester, UK.

Shen, L., Satta, G., & Joshi, A. K. (2007). Guided Learning for Bidirectional Sequence Classification. In *ACL 2007, Proceedings of the* 45^{th} *Annual Meeting of the Association for Computational Linguistics* (pp. 760–767). Prague, Czech Republic: The Association for Computer Linguistics.

Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn Treebank: An Overview. In *Treebanks: Building and Using Parsed Corpora* (pp. 5–22).

Tryggvason, A. (2009). *Named Entity Recognition for Icelandic.* Reykjavík University, Iceland. (BSc Thesis)

Zavrel, J., & Daelemans, W. (1999). Recent Advances in Memory-Based Part-of-Speech Tagging. In *VI Simposio Internacional de Comunicacion Social* (pp. 590–597). Santiago, Cuba.

Appendix A

Table A.1: The Icelandic tagset

Char#	Category/Feature	Symbol – semantics
1	Word class	**n**–noun
2	Gender	**k**–masculine, **v**–feminine, **h**–neuter, **x**–unspecified
3	Number	**e**–singular, **f**–plural
4	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
5	Article	**g**–with suffixed definite article
6	Proper noun	**m**–person name, **ö**–place name, **s**–other proper name
1	Word class	**l**–adjective
2	Gender	**k**–masculine, **v**–feminine, **h**–neuter
3	Number	**e**–singular, **f**–plural
4	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
5	Declension	**s**–strong declension, **v**–weak declension, **o**–indeclineable
6	Degree	**f**–positive, **m**–comparative, **e**–superlative
1	Word class	**f**–pronoun
2	Subcategory	**a**–demonstrative, **b**–reflexive, **e**–possessive, **o**–indefinite, **p**–personal, **s**–interrogative, **t**–relative
3	Gender/Person	**k**–masculine, **v**–feminine, **h**–neuter/**1**–1^{st} person, **2**–2^{nd} person
4	Number	**e**–singular, **f**–plural
5	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
1	Word class	**g**–article
2	Gender	**k**–masculine, **v**–feminine, **h**–neuter
3	Number	**e**–singular, **f**–plural
4	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
1	Word class	**t**–numeral
2	Category	**f**–alpha, **o**–numeric, **p**–percentage
3	Gender	**k**–masculine, **v**–feminine, **h**–neuter
4	Number	**e**–singular, **f**–plural
5	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
1	Word class	**s**–verb (except for past participle)
2	Mood	**n**–infinitive, **b**–imperative, **f**–indicative, **v**–subjunctive, **s**–supine, **l**–present participle
3	Voice	**g**–active, **m**–middle
4	Person	**1**–1^{st} person, **2**–2^{nd} person, **3**–3^{rd} person,
5	Number	**e**–singular, **f**–plural
6	Tense	**n**–present, **þ**–past
1	Word class	**s**–verb (past participle)
2	Mood	**þ**–past participle
3	Voice	**g**–active, **m**–middle
4	Gender	**k**–masculine, **v**–feminine, **h**–neuter
5	Number	**e**–singular, **f**–plural
6	Case	**n**–nominative, **o**–accusative, **þ**–dative, **e**–genitive
1	Word class	**a**–adverb and preposition
2	Category	**a**–does not govern case, **u**–exclamation, **o**–governs accusative, **þ**–governs dative, **e**–governs genitive
3	Degree	**m**–comparative, **e**–superlative
1	Word class	**c**–conjunction
2	Category	**n**–sign of infinitive, **t**–relative conjunction,
1	Word class	**e**–foreign word
1	Word class	**x**–unanalyzed word

REYKJAVÍK UNIVERSITY

HÁSKÓLINN Í REYKJAVÍK

School of Computer Science
Reykjavík University
Kringlan 1, IS-103 Reykjavík, Iceland
Tel: +354 599 6200
Fax: +354 599 6301
http://www.ru.is