Loftsson, Hrafn. 2008. Tagging Icelandic text: A linguistic rule-based approach. *Nordic Journal of Linguistics* **31.1**, XX-XX.

Tagging Icelandic text: A linguistic rule-based approach

Hrafn Loftsson

The Icelandic language is a morphologically complex language, for which a large tagset has been created. This paper describes the design of a linguistic rule-based system for part-of-speech tagging Icelandic text. The system contains two main components: a disambiguator, IceTagger, and an unknown word guesser, IceMorphy. IceTagger uses a small number of local elimination rules along with a global heuristics component. The heuristics guess the functional roles of the words in a sentence, mark prepositional phrases, and use the acquired knowledge to force feature agreement where appropriate. IceMorphy is used for guessing the tag profile for unknown words and for automatically filling tag profile gaps in the lexicon. Evaluation shows that IceTagger achieves 91.54% accuracy, a substantial improvement on the highest accuracy, 90.44%, obtained using three state-of-the-art data-driven taggers. Furthermore, the accuracy increases to 92.95% by using IceTagger along with two data-driven taggers in a simple voting scheme. The development time of the tagging system was only 7 man-months, which can be considered a short development period for a linguistic rule-based system.

Keywords data-driven tagging, disambiguator, linguistic rule-based tagging, simple voting, unknown word guesser

Hrafn Loftsson, School of Computer Science, Reykjavik University, IS-103 Reykjavik, Iceland.
hrafn@ru.is

1. INTRODUCTION

Part-of-speech (POS) tagging is the task of labelling words with the appropriate word class and morphological features. The string used as a label is called a tag, the set of labelling strings is called a tagset, and a program which performs tagging is called a tagger. Tagging text is needed for several natural language processing (NLP) tasks, e.g. grammar correction, syntactic parsing, information extraction, question-answering, and corpus annotation.

Since a word can be ambiguous in its POS (i.e. a word can have multiple possible tags), the main function of a tagger is to remove ambiguity. Many taggers perform this task by, first, introducing ambiguity (lexical phase) and, then, carry out disambiguation (disambiguation phase). The former, a relatively easy task, consists of introducing the ‘tag profile’ (the set of possible tags) for each word, both known and unknown words. This can be done with the help of a pre-compiled lexicon and an unknown word guesser, whose function is to guess the tag profile for words not known to the lexicon. The disambiguation task is more difficult, since, in order to disambiguate, a tagger needs to consider the context in which a particular word appears.

There are two main methodologies for disambiguation: the data-driven and the (handcrafted) linguistic rule-based approach. In the data-driven approach, a pre-tagged training corpus is used to automatically obtain information later to be used during disambiguation. The disambiguation information acquired can, for example, be in the form of statistics or rules. In contrast, a linguistic approach uses hand-crafted rules or constraints to eliminate inappropriate POS tags (or assign appropriate tags) depending on context.

There has been a tendency to develop data-driven taggers in the last ten to fifteen years. The data-driven taggers are language and tagset independent and usually simpler to develop than linguistic rule-based taggers. Additionally, developing a linguistic rule-based framework, able to compete with data-driven taggers, has been considered a difficult and time-consuming task (Brill 1992; Samuelsson 1994; Voutilainen 1995). A different opinion, indeed, has been expressed by Chanod & Tapanainen (1995).

A number of different data-driven tagging methods have been developed. Well known methods include, for example, probabilistic methods based on a Markov model (Kupiec 1992; Brants 2000) and maximum entropy (Ratnaparkhi 1996; Toutanova & Manning 2000), and the transformation-based learning approach (Brill 1992; Brill 1995). An accuracy of 96.5-96.7% has been achieved with these taggers for English text, using the Wall Street Journal corpus from the Penn Treebank (Marcus et al. 1994). A more recent result, showing accuracy of about 97.2% using the same corpus, has been obtained with dependency networks (Toutanova et al. 2003).

In contrast to data-driven taggers, linguistic rule-based taggers are developed with the purpose of tagging a specific language. One of the better known linguistic rule-based methods is the Constraint Grammar (CG) framework (Karlsson et al. 1995), in which both POS and grammatical functions are tagged. The English CG parser, EngCG, was ‘the first serious linguistic competitor to data-driven statistical taggers’ (Samuelsson & Voutilainen 1997).

It has been shown that combining taggers will often result in a higher tagging accuracy than achieved by individual taggers (Halteren et al.

2001). In a ‘simple voting’ combination scheme, each tagger gets an equal vote when voting for a tag and the tag with the highest number of votes is selected.

In this paper, I show that developing a linguistic rule-based system is a feasible option when tagging a morphologically complex language like Icelandic. I present a novel method for tagging Icelandic text which outperforms state-of-the-art data-driven methods. The tagger based on this method, hereafter called IceTagger, uses about 175 local rules for initial disambiguation and a set of heuristics (global rules), to force feature agreement where appropriate, for further disambiguation. By using IceMorphy, a new unknown word guesser, IceTagger’s accuracy on unknown words is higher than the corresponding accuracy in the data-driven taggers, and the overall tagging accuracy is 91.54% compared to 90.44% achieved by the best performing data-driven tagger. This amounts to 11.5% reduction in errors. Furthermore, by incorporating features of IceMorphy into the best performing data-driven tagger its accuracy increases from 90.44% to 91.18%. Finally, an accuracy of 92.95% is obtained, by using a simple voting scheme consisting of IceTagger and versions of two data-driven taggers using IceMorphy.

The development time of the tagging system (which is implemented in Java and includes a tokeniser, a sentence segmentiser, IceMorphy and IceTagger) was only 7 man-months. There are mainly two reasons for the short development time. First, the main lexicons used by the system are not extensive hand-crafted lexicons, but rather automatically generated from a corpus. Second, the emphasis is on using heuristics for disambiguation, as opposed to writing a large number of rules as is common in linguistic rule-based taggers. Taggers for other morphologically complex languages, in which feature agreement is important, might use similar heuristics for disambiguation.

The paper is organised as follows. Section 2 briefly describes the Icelandic tagset. Section 3 describes related tagging work. Section 4 and 5 describe the unknown word guesser and the tagger, respectively. Section 6 presents the evaluation results. In section 7, some methods of improving the accuracy of the data-driven taggers are discussed, and the paper concludes, in section 8, with directions for future work.

2. THE ICELANDIC TAGSET

Due to the morphological richness of the Icelandic language, the main tagset, created during the making of the ‘Icelandic Frequency Dictionary’ (IFD) corpus (Pind et al. 1991), is large (about 660 tags) and makes fine distinctions compared to related languages. Each character in a tag has a particular function. The first character denotes the word class. For each word class there is a predefined number of additional characters (at most six) which describe morphological features, like gender, number and case for nouns; degree and declension for adjectives; voice, mood and tense for verbs, etc.

Char #	**Category/ Feature**	**Symbol – semantics**
1	Word class	**n**-noun, **l**-adjective
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 (nouns)
5	Declension	**s**-strong, **v**-weak (adjectives)
6	Proper noun	**m**-person name, **ö**-place name, **s**-other
6	Degree	**f**-positive, **m**-comparative, **e**-superlative (adjectives)

Table 1. The semantics of the tags for nouns and adjectives.

Char #	**Category/ Feature**	**Symbol – semantics**
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**-1st person, **2**-2nd person, **3**-3rd person
5	Number	**e**-singular, **f**-plural
6	Tense	**n**-present, **þ**-past

Table 2. The semantics of the tags for verbs.

Table 1 shows the semantics of the tags for nouns and adjectives, and table 2 the semantics of the tags for verbs. To illustrate, consider the sentence *fallegu hestarnir stukku* ‘the beautiful horses jumped’. The corresponding tag for *fallegu* is *lkfnvf*, denoting adjective, masculine, plural, nominative, weak declension, positive; the tag for *hestarnir* is *nkfng*, denoting noun, masculine, plural, nominative with suffixed definite article; and the tag for *stukku* is *sfg3fþ* denoting verb, indicative mood, active voice, third person, plural, past tense.

Note the agreement in gender, number and case between the adjective and the noun, and the agreement in number between the subject and the verb (the verb also agrees with the subject in person, but since all nouns are third person by default, the person feature is not overtly expressed in this case).

Arguably, such a large and detailed tagset is not needed for all NLP applications. For example, features like the declension for adjectives, the type of proper nouns (which is indeed not of syntactic nature), mood and voice for verbs, and even the case feature might not be of interest for some applications. However, in this research the large tagset is (mainly) used, partly because of easier comparison with previous work, but mainly because a smaller version of the tagset has not yet been designed. On the other hand, such a detailed tagset may be necessary, for example, when constructing large tagged corpora to be used for linguistic research. Therefore, it is important that taggers, which can produce tags with detailed morphological information, are available for Icelandic.

3. RELATED WORK

3.1. Tagging methods

The earliest POS taggers used hand-crafted linguistic rules for assigning tags to words based on character affixes of words and on the basis of the tags of the surrounding words (Klein & Simmons 1963; Cherry 1980). The tagset used in the Cherry tagger was small (only 10 labels), since the purpose of the tagger was only to label each word with its word class. Tagging words with only the word class is in most cases of limited use for NLP applications. Therefore, it is common nowadays to use a tagset consisting of tens or hundreds of different tags.

In the past twenty years or so, the availability of larger tagged corpora, has encouraged researches to develop corpus-based or data-driven methods to tagging. Some of these methods have proven very effective for various languages, especially for the English language, which has been the main focus when applying these methods. It can be argued that to some extent the data-driven approaches to tagging have become a standard, and that the linguistic rule-based methods have become relatively infrequent.

The two approaches to tagging have often been considered ‘competing’ approaches in the literature (e.g. (Chanod & Tapanainen 1995; Samuelsson & Voutilainen 1997). It is, however, important to develop taggers for a particular language based on different approaches, because different approaches are likely to produce uncorrelated errors, which can be exploited in combination methods to yield better results (Halteren et al. 2001).

3.2. Data-driven tagging methods

Data-driven tagging methods use machine learning to automatically derive a language model from, usually, hand-annotated corpora. The main advantage with the data-driven approaches is that they are language independent and no (or limited) human effort is needed for derivation of the model. In what follows, we describe the data-driven methods (taggers) used in this research.[1]

One type of a data-driven tagger is a probabilistic trigram tagger based on a Markov model. A trigram tagger finds an assignment of POS to words by optimising the product of lexical probabilities and contextual probabilities. Lexical probability is the probability of observing word *i* given POS *j* ($p(w_i|t_j)$) and contextual probability is the probability of observing POS *i* given *k* previous POS ($p(t_i|t_{i-1},t_{i-2}, \ldots ,t_{i-k})$; k=2 for a trigram model). The probabilities of the model are estimated from a training corpus using maximum likelihood estimation. Due to data-sparseness problems, maximum likelihood estimation is generally combined with an appropriate smoothing method. After training, a new sentence can be tagged automatically by assigning it the tag sequence which receives the highest probability by the model. The TnT tagger (Brants 2000) is an example of an effective and an efficient trigram tagger.

Another type of a probabilistic data-driven tagger, the MXPOST tagger, is based on a maximum entropy approach (Ratnaparkhi 1996). It generates probability distributions like other statistical methods and, additionally, uses a binary feature representation to model tagging decisions which can be compared to rules in rule-based methods. A feature f_i asks a yes/no question about a particular history, h_i (a sequence of words and tags), available when predicting tag t_i. The feature f_i, which restricts the value of t_i, encodes any information that can be used to predict t_i, such as the spelling of the current word or the identity of tags and/or words. The goal of the model is to maximise the entropy of a distribution subject to certain feature constraints.

A third type of a successful data-driven tagger, the fnTBL tagger (Ngai & Florian 2001; Brill 1995), is based on the transformation-based learning approach. This approach is rule-based, but the transformation rules (which change a tag X to tag Y) are not hand-crafted, but rather automatically acquired from a pre-tagged corpus. The tagger initially assigns each word its most likely tag without regard to context.[2] At each iteration, the current assignment is compared to the pre-tagged text and a transformation is learnt which results in the greatest reduction of errors. The current assignment is updated using the learnt transformation before the next iteration starts. At the end, the result is an ordered list of transformations that can be applied to the output of the initial assignment to increase the accuracy. New unseen text is then tagged by first applying the initial annotator followed by the application of the ordered transformation rules.

The methods used by the three data-driven taggers for guessing unknown words are rather different. The unknown word guesser of the trigram tagger (TnT) uses ending analysis based on a probability distribution. The distribution for a particular ending is generated from words in the training corpus that share the same ending of some predefined length.

The maximum entropy tagger (MXPOST) uses feature templates when predicting tags for unknown words. The templates include prefixes and suffixes of length ≤ 4, as well as information regarding whether the word contains uppercase letters, hyphen(s) or a number.

The transformation-based tagger (fnTBL) uses a method, originally proposed by Brill (1995), which automatically learns cues to predict the most likely tag for unknown words. First, an unknown word is labelled as a proper noun if capitalised and a common noun otherwise. Secondly, transformation templates are used to learn rules which change the initial tag X to another tag Y. These templates make reference to any string of characters up to a bounded length and, typically, refer to suffixes or prefixes of words.

3.3. Linguistic rule-based tagging methods

In contrast to data-driven taggers, linguistic rule-based taggers are developed with the purpose of tagging a specific language. The purpose of the rules is either to assign tags to words depending on context or, in the more common reductionistic approach, to remove illegitimate tags from words based on context. The construction of a linguistic rule-based tagger can be a time-consuming task, since the rules are usually hand-crafted and the number of rules is often in the hundreds or thousands.

As an illustration of the number of rules used in linguistic rule-based taggers, one can mention the Swedish CG project where 2,100 rules are used to remove ambiguity (Birn 1998) and EngCG-2 (English CG version 2), a project developed over several years, with 3,600 rules (Samuelsson & Voutilainen 1997). A prerequisite for the effectiveness of a CG parser is the construction of a comprehensive lexicon and a morphological analyser, both of which demand large resources.

A disadvantage of the CG framework is ‘that constraints cannot be generalised, but have to be stated in a case by case fashion’ (Hinrichs &

Trushkina 2002). This is probably the reason for the large number of rules usually developed under this framework. In the German parsing scheme GRIP (Hinrichs & Trushkina 2002), two kinds of disambiguation rules are used: 'ordinary disambiguation rules', which eliminate readings for a single token based on local context, and general 'double reduction rules', which reduce readings of a sequence of tokens, e.g. inside noun phrases. IceTagger uses a similar idea, i.e. local rules for initial disambiguation and global rules or heuristics, to force feature agreement, for further disambiguation (see section 5).

The error rate of the EngCG-2 system has been reported as an order-of-magnitude lower than the error rate of a statistical tagger (Samuelsson & Voutilainen 1997). However, it is important to note that the EngCG-2 system does not perform full disambiguation and the results are, thus, only presented for the same ambiguity levels[3] in the two taggers. Hence, it is not clear what the difference in the error rate between the two systems would be if both would perform full disambiguation. Moreover, it can be inferred from this research that, when testing the EngCG-2 system, the unknown word ratio (which is not specified in the paper) is less than the corresponding ratio (2.01%) when testing the statistical tagger. The reason is that the former system uses a large hand-crafted lexicon, but the latter a lexicon derived from a training corpus.

3.4. Unknown word guessing

For a two-step tagger, as described in the introduction, the main problem in the lexical phase is guessing the tag profile for unknown words. Continuously extending the lexicon, to minimise the number of unknown words, is not practical because new words are constantly being introduced into a language. Therefore, in order to develop a high accuracy tagger, a good quality unknown word guesser is essential.

Most unknown word guessing modules use morphological/compound analysis or ending analysis, or a combination of both. The difference between morphological analysis and ending analysis is that the former bases its analysis on morphologically related words already known to the lexicon, whereas the latter bases its analysis solely on a word's ending. Not surprisingly, research has shown that morphological analysis is more accurate than ending analysis (Mikheev 1997; Nakov et al. 2003). This can be explained by the fact that morphologically related words share the same stem (the common part shared by all word forms) as the given unknown word, whereas ending analysis does not take the stem into account. Therefore, ending analysis, generally, produces more tag candidates than morphological analysis.

3.5. Tagging Icelandic

In 2002-2004, the Institute of Lexicography at The University of Iceland performed an Icelandic tagging experiment (Helgadóttir 2004) using three state-of-the-art data-driven taggers: fnTBL, MXPOST and TnT. The training data used is the IFD corpus, a balanced corpus consisting of about 590,000 tokens. 639 different tags appear in this corpus. Ten-fold cross validation[4] was used in the experiment. Each test corpus had, on the average, 59,030 tokens (3,691 sentences) and the average unknown word ratio was 6.84%. The highest accuracy, 90.36%, was obtained by the TnT tagger - the average results for the three taggers can be seen in table 3.

Words/Tagger	**fnTBL**	**MXPOST**	**TnT**
Unknown	54.03%	62.50%	71.60%
Known	91.36%	91.04%	91.74%
All	88.80%	89.08%	90.36%

Table 3. Average tagging accuracy in the Icelandic tagging experiment.

By using a simple voting scheme, i.e. selecting the tag chosen by two or more of the taggers and selecting TnT's tag in the case where all the three taggers disagreed, the total accuracy increased to 91.54%.

The accuracy of 90.36% in the Icelandic tagging experiment, obtained by the best performing single tagger, is considerably lower than the one achieved for related languages, e.g. Swedish where 93.55% accuracy was obtained in an experiment using the same taggers, a tagset consisting of 139 tags, and a training corpus of only 100k tokens (Megyesi 2002). This difference in accuracy can be explained by the large Icelandic tagset[5] as well as by the fact that Icelandic is morphologically more complicated than Swedish.

Computed using the IFD corpus, the average number of tags per token is 2.74, compared to, for example, 2.05 for Swedish text as determined by a lexicon used in a CG framework (Birn 1998). Using another frequently used criterion for ambiguity of a language, the ratio of ambiguous tokens in text to the total number of tokens, I found 59.7% of the tokens in the IFD corpus to be ambiguous. Compared to English, this is a much higher ratio where, for example, 35% of the word tokens in the Brown corpus were found to be ambiguous (Kupiec 1992). Note that the ambiguity rate and the ratio of ambiguous tokens are indeed tagset dependent as much as language dependent.

The difference in tagging accuracy between the three data-driven taggers can by and large be attributed to the difference in accuracy when tagging unknown words. The tagging accuracy of fnTBL for unknown words is relatively poor, about 25% less than the corresponding accuracy by TnT. On the other hand, since fnTBL achieves relatively high accuracy on known words, one can assume that with a better unknown word guessing module the total accuracy of fnTBL can be improved. Indeed, in section 7, I will show how to significantly improve fnTBL's unknown word tagging accuracy on Icelandic text.

It has been argued that although trigram taggers have performed well for English the same might not necessarily be true for other morphologically rich languages, for which large tagged corpora are not available (Schmid 1995). The problem is data sparseness, i.e. the size of the tagset in relation to the size of the training data. In the Icelandic tagging experiment, 639 tags occurred in the training corpus and, even though the size of the training data was moderately large or about 590,000 tokens, the tagset size is very large in relation to the size of the training data. 639 tags mean that 639^3 = 261 million contextual parameters need to be estimated for a trigram tagger. Hence, on the average, only about 0.002 tokens are available per parameter!

A linguistic rule-based tagger is not as sensitive to this data sparseness because its rules are not automatically derived from a tagged corpus, but rather hand-crafted using linguistic knowledge. Indeed, the rules are, to a certain extent, also dependent on a corpus - because they are constructed by examining phenomena extracted from it - but to a much lesser degree than a tagger based on a data-driven method. Moreover, some of the errors made by a data-driven tagger are due to the limited window size used for disambiguation (e.g. three words in the case of a trigram tagger). Contrastingly, a linguistic rule-based tagger can disambiguate focus words on the basis of words or tags which occur anywhere in the sentence, not only in the nearest neighbourhood.

Hence, I hypothesised that a carefully constructed linguistic rule-based tagger should perform better than a data-driven tagger when tagging a morphologically complex language, like Icelandic, using a large tagset.

Disambiguation rules can be developed using the fact that a limited set of word forms is responsible for a large part of the total ambiguity. Using the IFD corpus, I found that the 30 most frequent ambiguous word forms account for 50% of the total ambiguity, and the 153 most frequent ambiguous word forms are responsible for 67% of the total ambiguity.[6]

Interestingly, 21 out of the 30 most frequent word forms (i.e. 70%) are pure function words or pronouns, i.e. these words do not belong to any other word classes than adverbs, conjunctions, the infinitive marker, prepositions or pronouns. Using this knowledge, one can concentrate on writing disambiguation rules for the most frequent ambiguous word forms.

4. THE UNKNOWN WORD GUESSER

The unknown word guesser, IceMorphy, was designed to be a stand-alone module callable from different applications and to be used as an unknown word guesser in IceTagger. The purpose of IceMorphy is to generate the tag profile for a given word. It uses a familiar approach to unknown word guessing, i.e. it performs morphological analysis, compound analysis and ending analysis. Since morphological analysis (and compound analysis) is more accurate than ending analysis, it is important to design a system in which the main emphasis is on the former, and the latter is used when morphological analysis fails.

4.1. The morphological analyser

The morphological analyser tries to classify an unknown word as a member of a particular morphological class. A word belongs to a morphological class if the word's morphological ending harmonises with the inflection rules of the class. The current version uses 18 morphological classes for nouns, 5 classes for adjectives and 5 classes for verbs.

Basing the analysis on morphological classes is a common approach. It is, for example, used in two German morphological systems (Lezius 2000; Nakov et al. 2003) which use a previously compiled stem lexicon for lookup. However, the morphological analyser of IceMorphy is not dependent on a stem lexicon. Any general purpose lexicon, in which each word is tagged using the Icelandic tagset, will do, but the lexicon currently used is generated automatically from the IFD corpus.

For a given unknown word, *w*, a morphological class is guessed based on the morphological suffix of *w*. Then the stem *r* of *w* is extracted and all *k* possible morphological suffixes for *r* are generated, resulting in search strings, s_i (i=1,…,k), such that s_i=r+suffix$_i$. A lexicon lookup is performed for s_i until a word is found having the same morphological class (deduced from the tag profile for s_i) as was originally assumed or no match is found.

A similar approach has been taken when automatically inducing rules for unknown word guessing (Mikheev 1997), i.e. searching the lexicon for words that share the same stem, but have other morphological suffixes. However, the automatic rule induction method used to generate the rules for a tagger is very dependent on the training lexicon. Understandably, only rules that can be deduced by the lexicon will be produced. In contrast, the rules of IceMorphy are not dependent on a training lexicon, since they are compiled using linguistic knowledge. Another disadvantage with the automatic method is that it is not able to capture vowel mutation (e.g. *kökur* ‘cakes’ vs. *kak*a ‘cake’) which IceMorphy can do, because it is specifically tailored to Icelandic.

Consider the following example. Let us assume the word *hesturinn* ‘the horse’ (a masculine, singular, nominative case noun with a suffixed definite article) is an unknown word. Based on the suffix *urinn* the word is assumed to belong to the morphological class of regular masculine nouns. Consequently, the stem *hest* is extracted and all other inflectional endings for the stem are generated (*hest-ur*, *hest-*, *hest-i*, *hest-s*, *hest-ar*, *hest-a*, etc.). Hence, search strings, s_i, are generated, such that s_i=*hest*+suffix$_i$. A lookup is performed for each s_i, and if a lookup is successful for a given search string, the appropriate tag is returned.

Subsequently, modifications need to be performed on the tag returned. Let us assume, a lookup is successful for the search string *hesti* (s_3) whose corresponding tag is *nkeþ*. The fourth letter of a noun tag denotes the case and, since a nominative case for *hesturinn* was assumed, the tag *nkeþ* needs to be changed to *nken*. Furthermore, the fifth character for noun tags represents suffixed article and, therefore, the article letter (*g*) needs to be added to this tag, resulting in the correct tag *nkeng*.

The morphological analyser is, however, not flawless. To illustrate, consider what happens when analysing the word *búar* ‘neighbours’. Based on the morphological suffix *r* and the verb *búa* ‘to live’, the analyser classifies this word as a third (or second) person singular verb (similarity exists, for example, *ég borða* ‘I eat’, *hann borðar* ‘he eats’). Unfortunately, the third person singular form for this particular verb is *býr*, because the verb is irregular.

4.2. The compound analyser

This part of the unknown word guesser uses a straight-forward method of repeatedly removing prefixes from unknown words and performing a lookup for the remaining part of the word. If the remaining word part is not found in the lexicon, it is sent to the morphological analyser for further processing. If the lookup or morphological analysis deduces a tag *t* for the remaining word part, the original word (without prefix removal) is given the same tag *t*. To illustrate, consider the compound word *nýfæddur* ‘newborn’. By removing the first two letters, *ný* ‘new’, a lexicon lookup is performed for the substring *fæddur*. If *fæddur* is found in the lexicon with tag *t*, the word *nýfæddur* is assigned the same tag *t*. Otherwise, *fæddur* is sent to the morphological analyser for further processing.

As the morphological analyser, the compound analyser can make mistakes. Consider, for example, the past participle *upprisinn* ‘risen up’. The compound analyser will remove the prefix *upp* and perform a lookup for the remaining word *risinn*, which, incidentally, is the masculine, singular, nominative noun *risi* ‘a giant’ with a suffixed definite article. Thus, the analyser incorrectly classifies the word *upprisinn* as a noun.

4.3. The ending analyser

The ending analyser is called if an unknown word can be deduced neither by morphological analysis nor by compound analysis. This component uses a hand-written endings lexicon along with an automatically generated one. The former, which is consulted first, is mainly used to capture common endings for verbs and adjectives, for which numerous tags are possible. By only using an automatically generated list of endings from a tagged corpus, it is almost certain (unless the tagged corpus is enormous) that not all possible tags for given adjectives will be deduced.

The automatically generated ending lexicon is constructed in the following manner. From a tagged corpus, all possible word endings of length 1 to 5 are collected together with the corresponding tags (the minimum length of the remaining substring is 2 characters). I assume that the endings are different for capitalised words vs. other words and therefore produce two endings lexicons, one for proper nouns and another for all other words. Endings that appear with less frequency than some specific threshold (10 in my case) are filtered out.

As pointed out earlier, ending analysis is less accurate than morphological/compound analysis. For example, for the word *bleðillinn* ‘the sheet’, the ending analyser proposes the four tags *nkeng_nkeog_lkensf_lkeosf*, based on the *llinn* ending. However, only the first tag is correct for this particular word.

4.4. Tag profile gaps

An important feature of IceMorphy is its handling of tag profile gaps. A tag profile gap arises when a particular word, listed in the lexicon, has some missing tags in its set of possible tags. This, of course, presents problems to a disambiguator, since its purpose is to select one single correct tag from all possible ones. For each noun, adjective or verb of a particular morphological class, IceMorphy uses the methods described above to fill in the gaps for the given word.

To illustrate, consider the word *konu* ‘woman’, and let us assume that only the tag *nveo* (denoting noun, feminine, singular, accusative) is found in the lexicon. Based on the *u* morphological suffix and the accusative case of the tag, IceMorphy assumes the word belongs to a particular morphological feminine noun class, in which singular accusative, dative and genitive cases have the same word form. Consequently, IceMorphy generates the correct missing tags: *nveþ* and *nvee*.

5. THE RULE-BASED TAGGER

The main characteristic of the disambiguation part of IceTagger is the use of only a small number of local rules (about 175) along with heuristics that perform further global disambiguation based on feature agreement.

5.1. Idioms and phrasal verbs

The first step of the disambiguation process is to identify idioms, i.e. bigrams and trigrams which are always tagged unambiguously. Idioms are identified by examining lexical forms of adjacent words. The list of idioms was constructed semi-automatically in the following manner. First, we extracted automatically all trigrams in the IFD corpus that occurred at least ten times with the same tag sequence. Additionally, we hand-constructed a list of frequently occurring bigrams tagged unambiguously, by examining the development corpus (described in section 6.2).

The second step is to identify phrasal-verbs, whose words are adjacent in text. An Icelandic phrasal verb is a verb-particle pair (like *fara út* ‘go out’) where the particle is an adverb (because it is associated with a particular verb), but not a preposition. An automatically generated lexicon (from the IFD corpus) is used for recognising phrasal verbs.

5.2. Local rules

The third step of the disambiguation process is the application of local elimination rules which perform disambiguation based on a local context (a window of 5 words; two words to the left and two words to the right of the focus word). The purpose of a local rule is to eliminate inappropriate tags from words. This reductionistic approach is common in rule-based taggers, and is, for example, used in the CG systems.

In principle, the local rules are unordered. The firing of a rule is, however, dependent on the order of the words in a sentence. A sentence to be tagged is scanned from left to right and all tags of each word are checked in sequence. Depending on the word class (the first letter of the tag) of the focus word, the token is sent to the appropriate disambiguation routine which checks a variety of disambiguation constraints applicable to the particular word class and the surrounding words. At each step, only tags for the focus word are eliminated.

The rules are written in a separate file. A Java-like syntax is used and the rules are compiled to Java code. The format of a local rule is:

RULE <condition>;

A <condition> is a boolean expression, whose individual components can refer to lexical forms or individual characters (word class/morphological features) of tags. If <condition> is true, then the tag in question for the focus word is eliminated. The following are examples of *<condition>* (L_1/R_1 and L_2/R_2 denote tokens one and two to the left/right of the focus word, *F*, respectively):

L_1.*isOnlyWordClass(x)* AND L_2.*isOnlyWordClass(y)*
R_1.*isWordClass(x)* OR R_2.*isWordClass(y)*
L_1.*isWordClass(x)* AND t.*isCase(y)* AND *t.isGender(z)*
R_1.*lexeme.equals(x)* AND F.*isWordClass(y)*

A predicate like *isOnlyWordClass(x)* is true for a word belonging only to the word class *x*, while a predicate like *isWordClass(x)* is true for a word having the word class *x* as one of its possible word classes.

To exemplify, consider the following sentence: *við vorum alltaf ein* 'we were always alone'. The word *við* can have the following five tags: *ao_aþ_fp1fn_aa_nkeo*. These tags denote a preposition which governs accusative, a preposition which governs dative, a first person, plural, nominative personal pronoun, an adverb, and a masculine, singular, accusative noun, respectively. Since the following word is a verb, *vorum*, and prepositions only precede nominals, a rule (for prepositions), with *<condition> = R_1.isOnlyWordClass(Verb)*, eliminates preposition tags in this context, leaving only the three tags *fp1fn_aa_nkeo*.

5.3. Heuristics

Once local disambiguation has been carried out, each sentence is sent to a global heuristic module. Its purpose is to perform grammatical function analysis, guess prepositional phrases, and use the acquired knowledge to force feature agreement where appropriate. I call these heuristics global because, when disambiguating a particular word, a heuristic can refer to a word which is not in the nearest neighbourhood. The heuristics are a collection of algorithmic procedures that guess the syntactic structure/functions of the sentence and use it as an aid in the disambiguation process. Similar heuristics to those described below may be applicable to other morphologically complex languages.

Before the heuristics are applied, each sentence is partitioned into clauses using tokens like a comma, a semicolon, a coordinating conjunction, and a relativizer as separators (care is taken not to break up enumerations into individual parts). The heuristics then repeatedly scan each clause and perform the following in order:

1. Mark prepositional phrases
2. Mark verbs
3. Mark subjects of verbs
4. Force subject-verb agreement
5. Mark objects of verbs
6. Force subject-object agreement
7. Force verb-object agreement
8. Force agreement between nominals
9. Force prepositional phrase agreement

A detailed description and evaluation of all the heuristics can be found in Loftsson (2006a), but here I only discuss, in detail, the functionality of the heuristic which guesses the subject of a verb (heuristic 3). This heuristic assumes that verbs in the current clause have already been marked by heuristic 2. For a given verb *v*, the tokens are first scanned starting from the left of *v* (since SVO order is the most likely one). If the immediate token to the left of *v* is a relativizer or a comma, then it is assumed that the subject can be found in the previous clause (see below). Otherwise, if the current token is a nominal and it agrees with *v* in person and number, it is marked as a subject - if not, the scanning continues. If no subject candidate is found to the left of *v*, a search continues using the next two tokens to the right of *v* (it is thus assumed that subjects appearing further away to the right are unlikely), using the same feature agreement criterion as before. If at this point no subject candidate has been found, a search is performed in the previous clause, and the first nominal found is then marked as the subject (if it is not already marked as an object of a verb in the previous clause).

To illustrate the effect of all the above heuristics, consider the sentence *ég fór svartar götur í vesturátt* 'I went dark streets in west-direction'.

After the application of local rules, the word/tag(s) pairs are the following:

*ég/**fp1en** fór/**sfg3eþ_sfg1eþ** svartar/**lvfosf_lvfnsf** götur/**nvfo_nvfn** í/**aþ_ao** vesturátt/**nveo_nveþ***

Heuristic 1 starts by marking *í vesturátt* as a prepositional phrase, since *í* is a preposition and *vesturátt* is a nominal which agrees in case with the preceding preposition.

Then the main verb is marked by heuristic 2 (*fór* is the only possible verb).

Heuristic 3 marks the 1st person pronoun *ég* as a subject and heuristic 4 removes the 3rd person tag from *fór,* since the subject is a 1st person (leaving only the tag *sfg1eþ*).

Thereafter, heuristic 5 marks *götur* as the object and heuristic 7 removes the nominative tag *nvfn* from *götur*, because the verb *fór* demands an accusative object (this information is obtained from one of the lexicons described below).

Heuristic 8 then removes the nominative tag *lvfnsf* from the adjective *svartar* because of feature agreement with the already disambiguated noun *götur*.

Finally, heuristic 9 removes the dative tag *aþ* from the preposition *í* and the dative tag *nveþ* from the nominal *vesturátt* because the verb-preposition pair *fór-í* governs accusative case (again this information is obtained from one of the lexicons described below).

The correct tag sequence, *ég/**fp1en** fór/**sfg1eþ** svartar/**lvfosf** götur/**nvfo** í/**ao** vesturátt/**nveo***, is thus produced.

To give one example where long-range feature agreement comes into play, consider the sentence *ég reis á fætur og gekk að glugganum* ‘I got up and walked towards the window’. Possible tags for the verb *gekk* are *sfg3eþ_sfg1eþ*, e.g. indicative, active, past tense, 3rd or 1st person singular. The 1st person tag, *sfg1eþ* is the correct tag in this case, because the only subject in the sentence is the 1st person pronoun *ég* at the start of the sentence. The above heuristics will correctly tag a scenario like this.

Two automatically generated lexicons are used as an aid in prepositional phrase agreement and verb-object agreement. The former includes information about the case a particular verb-preposition pair demands (a particular preposition can govern more than one case in Icelandic, depending on the preceding verb), whereas the latter states the case that an object of a transitive verb should have.

In addition to heuristics 1-9, specific heuristics are used to choose between supine and past participle verb forms, and ensuring agreement between reflexive pronouns and their antecedents. If a word is still ambiguous after the application of the heuristics, the default heuristic is simply to choose the most frequent tag according to frequency information derived from the IFD corpus.

Since the local rules are relatively few, the global heuristics have a large impact on the overall performance of IceTagger. For one of the test corpora, the overall tagging accuracy increases from 84.91% (after the application of local rules) to 91.36% when global heuristics are used (when computing these figures, I select the most likely tag for words that are still ambiguous after application of local rules or heuristics).

Note that due to the nature of the heuristics, IceTagger could be used to generate a kind of a shallow parsing output. Prepositional phrases are marked specifically and noun phrases could be marked as well (in connection with heuristic 8). Moreover, subjects and objects are tagged explicitly.

5.4. Special verbs

Some Icelandic verbs have special characteristics. IceTagger keeps auxiliaries, the verbs *vera/verða* 'be/become', and verbs that demand non-nominative case subjects in a special (base) lexicon (all possible word forms for these verbs are included in the lexicon). These verbs are marked in a special way to facilitate correct disambiguation decisions when encountered.

The special marking, for example, includes information on which case a verb demands for its (non-nominative case) subject. As another example, tags for word forms for the verbs *vera/verða* are marked with a special code, which tells the tagger to expect a predicative nominative for this verb. This mechanism simplifies rule writing, because rules do not need to be written for each word form of the given verb. Instead the rules can refer to the special codes included in the corresponding tags.

6. EVALUATION

6.1. IceMorphy

I first evaluated the unknown word guesser, IceMorphy. I hand-tagged 400 unknown words randomly extracted from one of the test corpora. The number of relevant (correct) tags was 1194, but IceMorphy generated 1554 tags, which divided in the following manner: 555 (35.71%) common noun tags, 266 (17.12%) proper noun tags, 553 (35.59%) adjective tags, 170 (10.94%) verb tags and 10 (0.64%) tags belonging to other word classes.

Table 4 shows precision and recall for the main word classes and for all tags as a group. Precision and recall are defined in the following manner:

precision = # of relevant generated tags / # of generated tags
recall = # of relevant generated tags / # of relevant tags

	Common nouns	**Proper nouns**	**Adjectives**	**Verbs**	**All words**
Precision	64.32%	18.80%	72.73%	52.35%	58.17%
Recall	83.80%	58.14%	71.43%	82.41%	75.71%

Table 4. Precision and recall for the given word classes using IceMorphy on 400 randomly selected words.

For the purpose of tagging, high recall is more important than high precision. It is imperative that an unknown word guesser produces as many relevant tags as possible for a tagger to disambiguate. If a relevant tag is missing, a tagger may not be able to disambiguate correctly.

Let us assume that the figures in table 4 represent the precision and recall figures for the whole population of unknown words analysed by IceMorphy. Based on this assumption, the recall figures place an approximate upper bound on the tagging accuracy obtainable by IceTagger for unknown words. For example, the figures indicate that the overall tagging accuracy for unknown words of IceTagger will not exceed 75.7%, and that the tagger will produce highest accuracy figures for nouns. Indeed, in the next section, it will be shown that this upper bound holds for all classes except adjectives, for which the actual tagging accuracy is, in fact, higher than 71.4%.

The low precision and recall for proper nouns can be explained by the following. The last character of the tag for proper nouns denotes named entity information, i.e. a person name, place name or any other name. Since IceMorphy does not include a named entity recogniser, it has difficulty guessing the correct tag for proper names. When the last character of proper nouns is ignored, precision and recall for proper nouns increases to 31.20% and 96.51%, respectively.

No morphological analyser for the Icelandic language has previously been published and, thus I can not compare my figures to results published for Icelandic.[7] Additionally, it is difficult to do comparison across languages because of different levels of morphological complexity and, hence, different tagsets. For the sake of doing one comparison, the morphological classifier of unknown German nouns described by Nakov et al. (2003) achieves 82%-89% recall (depending on the test corpus). The purpose of the German analyser is to guess morphological classes using a comprehensive stem lexicon, as opposed to predicting individual tags using a lexicon derived from a tagged corpus, as is the case for IceMorphy. Since using a more comprehensive lexicon will most certainly increase IceMorphy's accuracy, I believe my guesser obtains good results.

6.2. IceTagger

In order to make a fair comparison between IceTagger and the three data-driven taggers, I used exactly the same training and test corpora as were used in the Icelandic tagging experiment (described in section 3.5). Recall that the available hand-tagged corpus is the IFD corpus consisting of about 590,000 tokens. Pairs of 10 training corpora (each containing 90% of the IFD corpus) and 10 test corpora (each containing 10% of the IFD corpus) were constructed, in the Icelandic tagging experiment. The test corpora are independent of each other, whereas the training corpora overlap. I used the first nine of these test corpora for evaluation, but the tenth one was used as a development corpus.

For each test corpus the corresponding training corpus was used to deduce the lexicons used by IceTagger - the main lexicon stating word forms and allowable tags (55,600 word forms, on the average), the lexicon for phrasal verb recognition, the lexicon for verb-preposition pairs and the lexicon for verb case governance. Since each main lexicon was only deduced from the corresponding training corpus, but not as well from the test corpus, unknown words in the Icelandic tagging experiment are also unknown in tests of IceTagger. There is, however, one exception. IceTagger uses an additional base lexicon which includes words of the closed word classes (pronouns, prepositions, conjunctions) and about 120 irregular verbs. Since these words are, indeed, very common and are therefore, in most cases, already in the lexicon derived from a training corpus, the ratio of unknown words, when testing IceTagger, is only a fraction lower than the corresponding ratio in the Icelandic tagging experiment (6.79% vs. 6.84%, on the average). Table 5 shows average figures for the nine test corpora used in my experiment.

Before I present the results, let us first discuss baseline accuracy figures for tagging Icelandic text using the given tagset. By baseline accuracy, I mean the lower bound on the accuracy that a tagger should achieve. A naive baseline tagger can be constructed by always assigning each known word its most frequent tag, and assigning each unknown word starting with a lower/upper case letter the most frequently occurring tag for common/proper nouns. The average baseline tagging accuracy for the nine test corpora using this method is 76.27% (see table 5). This figure is considerably lower than the baseline accuracy of 80.75% for Swedish (Megyesi 2002). Using similar methods for known words and assigning unknown words the most common tag for words ending in the same three letters, a baseline tagging accuracy of around 92% has been reported for English (Brill 1992). As before, keep in mind that these figures depend on the tagset used as much as on the language.

					Baseline accuracy		
# of tokens	# of sent.	unkn. ratio	ambig. words	ambig. rate	unkn. words	known words	all words
59,081	3,684	6.79%	60.43%	2.76	4.39%	81.84%	76.27%

Table 5. Average figures for the nine test corpora used for evaluation.

Let us now consider the results achieved by IceTagger in comparison to the three data-driven taggers: MXPOST, fnTBL and TnT. IceTagger achieves superior accuracy figures for unknown words, known words and all words for all the nine test corpora. For each test corpus, the difference between IceTagger and the closest competitor (TnT) is statistically significant ($\alpha<0.005$, using McNemar's chi-squared goodness-of-fit test as described by Dietterich (1998)). The average overall tagging accuracy is 91.54% compared to 90.44% for the TnT tagger. Three out of every four unknown words receive the correct analysis by IceTagger. Overall, according to the results, IceTagger makes 11.5% less errors than the TnT tagger. The average results using the nine test corpora can be seen in table 6.[8]

Words/Tagger	MXPOST	fnTBL	TnT	IceTagger
Unknown	62.29%	55.51%	71.68%	75.09%
Known	91.00%	91.82%	91.82%	92.74%
All words	89.03%	89.33%	90.44%	91.54%

Table 6. Average tagging accuracy of IceTagger in comparison to the three data-driven taggers.

	Common nouns	Proper nouns	Adjectives	Verbs	All words
Accuracy	81.13%	56.75%	75.73%	69.90%	75.09%

Table 7. Average tagging accuracy of IceTagger for different word classes of unknown words.

Table 7 shows the average tagging accuracy of IceTagger for different word classes of unknown words. Recall that the recall figures for the word classes shown in table 4 constitute an approximate upper bound on the figures shown in table 7.

I have previously stated that morphological/compound analysis is more accurate than ending analysis. This fact is reflected in the accuracy of unknown words in IceTagger using different components of IceMorphy; see table 8. On the average, the morphological analyser produces results for 40.22% of unknown words and the average tagging accuracy for morphologically analysed words is 84.74%. In contrast, the ending analyser produces analysis for 26.29% of unknown words, resulting in only 57.09% average tagging accuracy. Bear in mind that the ending analyser is only applied if morphological/compound analysis fails, and therefore, to some extent, handles the more difficult cases.

Component of IceMorphy			**Accuracy in IceTagger**		
Morpho-logical	**Compound**	**Ending**	**Morpho-logical**	**Compound**	**Ending**
40.22%	31.10%	26.29%	84.74%	78.57%	57.09%

Table 8. Ratio of unknown words that are successfully analysed by components of IceMorphy, and the corresponding tagging accuracy of IceTagger for these words.

IceTagger has a number of different components, which have been described above. In order to estimate how individual components contribute to the overall accuracy, I evaluated different versions of IceTagger with one (or two) component(s) removed in each version. The first version uses only ending analysis (i.e. morphological and compound analysis is removed) for unknown word guessing, the second version does not use tag profile gap filling, the third version does not include the idiom module, and in the fourth version the module for phrasal verbs has been removed. Table 9 shows the result.

Words/Tagger	**Only ending analysis for unknown words**	**No tag profile gap filling**	**No idiom module**	**No phrasal verb module**
Unknown	62.84%	74.37%	75.07%	75.09%
Known	92.58%	91.88%	92.48%	92.66%
All words	90.56%	90.69%	91.30%	91.47%

Table 9. Average tagging accuracy for different versions of IceTagger.

From these results, we can deduce the following. First, the morphological/compound analysis module of IceMorphy is important, because, without it, the accuracy for unknown words drops from 75.09% to 62.84%. Furthermore, the tagging accuracy of unknown words analysed by the ending analyser increases when it is allowed to process all cases, but not only those for which morphological/compound analysis fails. Second, the average contribution of the tag profile gap filling module for all words is 0.85%. Third, the average contribution of the idiom module is 0.24%, but the contribution of the phrasal verb module is relatively modest.

In an evaluation of tagging accuracy it is common to present figures for the somewhat unrealistic case of a closed vocabulary, i.e. assuming no existence of unknown words. When I evaluated IceTagger under this assumption, the average accuracy was 93.84%.

Earlier, I explained the large difference in tagging accuracy when tagging Icelandic text vs. English text by the size of the Icelandic tagset used. Accordingly, one would expect the tagging accuracy of Icelandic to get closer to the accuracy published for English when condensing the Icelandic tagset. I tested this by ignoring some of the morphological features (like the case feature) when calculating the accuracy, effectively resulting in a set of 99 tags.[9] Indeed, table 10 shows that when using a much smaller tagset the highest tagging accuracy for all words approaches 96%. Moreover, the TnT tagger obtains higher accuracy than IceTagger for this smaller tagset, which seems to indicate that IceTagger is more suitable for applications that need more fine-grained morphological information. Note, however, that valuable information (for example, indicating syntactic functions) is lost if the case feature is ignored.[10]

Words/Tagger	**MXPOST**	**fnTBL**	**TnT**	**IceTagger**
Unknown	81.76%	76.15%	87.49%	86.51%
Known	95.79%	96.33%	96.34%	96.25%
All words	94.83%	94.95%	95.74%	95.59%

Table 10. Average tagging accuracy using a condensed tagset of 99 tags.

7. DISCUSSION

As stated above, the tagging accuracy of IceTagger is statistically significantly better than the accuracy obtained by the three data-driven taggers. IceTagger makes 11.5% less errors than the best performing data-driven tagger, TnT.

The following question now comes to mind: Could IceTagger's accuracy have been obtained by improving the data-driven taggers?

Recall that the unknown word accuracy of the fnTBL tagger is relatively low and an improvement should be possible. To verify this, I overwrote fnTBL's default initial tagging assignment for unknown words by calling IceMorphy instead (this was possible due to the availability of fnTBL's source files). I made IceMorphy return the most probable tag for each unknown word, since a transformation-based method needs one single tag for its initial assignment. This significantly increased the accuracy of fnTBL (I call this tagger fnTBL*). The unknown word accuracy/overall accuracy increased from 55.51%/89.33% (see table 6) to 66.30%/90.15% (see table 11).

As stated earlier, the tag profile gap filling component of IceMorphy contributes about 0.85% to the total average accuracy of IceTagger. Therefore, I assumed that TnT's tagging accuracy would get closer to IceTagger's accuracy when using tag profile gap filling. I tested this hypothesis in the following way. Each record in the lexicon used by TnT consists of a word and the corresponding tags found in the training corpus. Additionally, to facilitate lexical probability calculations, each tag is marked by its frequency (i.e. how often the tag appeared as a label for the given word). Using the same training corpus, I made IceMorphy generate a 'filled' lexicon such that each generated missing tag was marked with the frequency 1.[11] Indeed, by running TnT with this extended lexicon (I call this tagger TnT*) I obtained an overall average tagging accuracy which is about 0.35% less than obtained by IceTagger (see table 11).[12]

I have shown that by using features of IceMorphy, the tagging accuracy of fnTBL and TnT could be improved, and, indeed, get quite close to IceTagger's tagging accuracy. This shows that IceMorphy is a critical component for improving tagging accuracy of Icelandic text. It is, however, important to keep in mind that the tagging accuracy of IceTagger can still be improved by refining each of its individual components. For example, writing more local rules, to handle the frequent ambiguous word forms, will certainly be beneficial. On the other hand, improving the accuracy of a data-driven tagger is likely to be a more difficult task, since it is not tailored to a specific language.[13]

Words/Tagger	**fnTBL***	**TnT***	**IceTagger**	**Simple voting**
Unknown	66.30%	72.75%	75.09%	76.57%
Known	91.90%	92.53%	92.74%	94.15%
All words	90.15%	91.18%	91.54%	92.95%

Table 11. Tagging accuracy using features of IceMorphy.

The importance of IceMorphy is even more evident when the three taggers, fnTBL*, TnT* and IceTagger, are combined using the following simple voting scheme. A tag agreed upon by at least two of the taggers is chosen. In the case where all the taggers disagree the tag proposed by IceTagger is selected. By using this voting scheme the overall tagging accuracy increases to 92.95% (see table 11). This is a large improvement from the previously reported 91.54% voting scheme result, obtained by combining 'raw' versions of fnTBL, MXPOST and TnT (Helgadóttir 2004).

8. CONCLUSION

In this paper, I have presented a linguistic rule-based tagger, IceTagger, which, together with an unknown word guesser, IceMorphy, obtains higher tagging accuracy than state-of-the-art data-driven taggers, when tagging running Icelandic text using a large tagset.

The disambiguator uses only about 175 local rules, but is able to achieve high accuracy through the use of global heuristics along with automatic tag profile gap filling. The heuristics guess the functional roles of the words in a sentence, mark prepositional phrases, and use the acquired knowledge to force feature agreement where appropriate. Other morphologically complex languages might use similar heuristics for POS tagging.

This work shows that a linguistic rule-based approach does not have to be very labour intensive in order to achieve high tagging accuracy. The main lexicons used by the system are automatically derived from a tagged corpus. In the design of the disambiguation phase of IceTagger, main emphasis was put on developing the heuristics, instead of writing a large set of constraint rules. This is the main reason for only 7 man-months development time of the system.

The average tagging accuracy of IceTagger is 91.54% compared to 90.44% for the best performing data-driven tagger, TnT, using the same test corpora. By using the tag profile gap filling mechanism of IceMorphy, I was able to increase the accuracy of TnT to 91.18%. Moreover, by combining IceTagger with versions of fnTBL and TnT, which use features of IceMorphy, the tagging accuracy increased to 92.95%.

In future work, I would like to improve individual components of IceTagger and experiment with different combination methods with the purpose of increasing the accuracy further.[14] Moreover, a smaller version of the tagset should be designed and the taggers re-evaluated using the new tagset.

ACKNOWLEDGEMENTS

I would like to thank the Institute of Lexicography at the University of Iceland for kindly providing access to the training and test corpora used in this research. Moreover, I thank Professor Yorick Wilks and Dr. Mark Hepple for valuable comments and suggestions in the preparation of this paper; any mistakes are my own. Lastly, I thank the three anonymous reviewers for many helpful comments which improved this paper.

REFERENCES

Birn, Juhani. 1998. Swedish Constraint Grammar: A short presentation. Technical Report, Lingsoft, Inc.

Brants, Thorsten. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied natural language processing*. Seattle, WA, 224-231.

Brill, Eric. 1992. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the 3rd Conference on Applied natural language processing*. Trento, 152-155.

Brill, Eric. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics* **21(4),** 543-565.

Chanod, Jean-Pierre & Pasi Tapanainen. 1995. Tagging French – comparing a statistical and a constraint-based method. In *Proceedings of the 7th Conference on European Chapter of the ACL*. Dublin, 149-156.

Cherry, Lorinda L. 1980. PARTS - A System for Assigning Word Classes to English Text. Technical Report, Computing Science #81, Bell Laboratories.

Dietterich, Thomas G. 1998. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, **10(7),** 1895-1924.

Halteren, Hans van, Jakub Zavrel & Walter Daelemans. 2001. Improving Accuracy in Wordclass Tagging through Combination of Machine Learning Systems. *Computational Linguistics*, **27(2)**, 199-230.

Helgadóttir, Sigrún. 2004. Testing Data-Driven Learning Algorithms for PoS Tagging of Icelandic. In Henrik Holmboe (ed.), *Nordisk Sprogteknologi 2004*. Copenhagen: Museum Tusculanums Forlag, 257-265.

Hinrichs, Erhard W. & Julia Trushkina. 2002. Getting a Grip on Morphological Disambiguation. In *Proceedings of KONVENS 2002, 6. Konferenz zur Verarbeitung natürlicher Sprache*. Saarbrücken, 59-66.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä & Arto Anttila (eds.). 1995. *Constraint Grammar*. Berlin: Mouton de Gruyter.

Klein, Sheldon & Robert Simmons. 1963. A computational approach to grammatical coding of English words. *Journal of the ACM*, **10**, 334-347.

Kupiec, Julian. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech and Language*, **6**, 225-242.

Lezius, Wolfgang. 2000. Morphy - German Morphology, Part-of-Speech Tagging and Applications. In *Proceedings of the 9th EURALEX International Congress*. Stuttgart, 619-623.

Loftsson, H. 2006a. Tagging a morphologically complex language using heuristics. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo & Tapio Pahikkala (eds.), *Advances in Natural Language Processing, 5th International Conference on NLP, FinTAL 2006, Proceedings*. Turku, 640-651.

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

Marcus, Mitchell P., Beatrice Santorini & Mary A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **19(2),** 313-330.

Megyesi, Beáta. 2002. *Data-driven Syntactic Analysis: Methods and applications for Swedish*. Ph.D. dissertation. KTH, Stockholm.

Mikheev, Andrei. 1997. Automatic Rule Induction for Unknown Word Guessing. *Computational Linguistics*, **21(4)**, 543-565.

Nakov, Preslav, Yury Bonev, Galia Angelova, Evelyn Cius & Walther von Hahn. 2003. Guessing Morphological Classes of Unknown German Nouns. In *Proceedings of Recent Advances in Natural Language Processing*. Borovets, 347-356.

Ngai, Grace & Radu Florian. 2001. Transformation-Based Learning in the Fast Lane. In *Proceedings of the 2nd Conference of the North American Chapter of the ACL*. Pittsburgh, PA, 1-8.

Pind, Jörgen, Friðrik Magnússon & Stefán Briem. 1991. *The Icelandic Frequency Dictionary*. Reykjavik: The Institute of Lexicography, University of Iceland.

Ratnaparkhi, Adwait. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Philadelphia, PA, 133-142.

Samuelsson, Christer. 1994. Morphological tagging based entirely on Bayesian inference. In Robert Eklund (ed.), *9th Nordic Conference on Computational Linguistics*. Stockholm, 225-238.

Samuelsson, Christer & Atro Voutilainen. 1997. Comparing a Linguistic and a Stochastic Tagger. In *Proceedings of the 8th Conference on European Chapter of the ACL*. Madrid, 246-253.

Schmid, Helmut. 1995. Improvements in Part-of-Speech Tagging with an Application to German. In Helmut Feldweg & Erhard W. Hinrichs (eds.), *Lexikon und Text*. Tübingen: Max Niemeyer Verlag, 47-50.

Toutanova, Kristina & Christopher D. Manning. 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*. Hong Kong, 63-70.

Toutanova, Kristina, Dan Klein, Christopher D. Manning & Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the 2003 Conference of the North American Chapter of the ACL on Human Language Technology*. Edmonton, 252-259.

Voutilainen, Atro. 1995. A syntax-based part-of-speech analyzer. In *Proceedings of the 7th Conference on European Chapter of the ACL*. Dublin, 157-164.

NOTES

[1] For other state-of-the-art taggers, consult, for example, Toutanova et al. (2003).
[2] Thus, a transformation-based tagger is an example of a tagger which does not start by introducing the whole tag profile for each word.
[3] An ambiguity level is a predefined ambiguity rate (average number of tags per word).
[4] The corpus is randomly divided into ten disjoint subsets of (approximately) equal size. The training is performed ten times on nine of these ten disjoint datasets and then testing is performed on the one left out, each time leaving out a different one.
[5] Intuitively, a larger tagset should result in lower tagging accuracy, since, for a larger tagset, the tagger simply has more tags to choose from for each word. In a tagging experiment on Dutch text, using a tagset of 341 tags, 92.06% accuracy was achieved by the TnT tagger (Halteren et al. 2001).
[6] Total ambiguity = $\sum_{i=1}^{n} freq(w_i) * tags(w_i)$, where *freq(w$_i$)* is the frequency of w_i in the given corpus and *tags(w$_i$)* is the number of possible tags for w_i. Only ambiguous words are taken into account.
[7] The morphological analyser *Púki* is a spelling tool developed by a private Icelandic software company. The software is proprietary, intended to be used with Microsoft Office, and accuracy figures for the analyser have not been published.
[8] When we repeated the Icelandic tagging experiment (Helgadóttir 2004), we obtained slightly different results for the fnTBL and the TnT tagger. The difference can be explained by the fact that our training corpus had empty lines between sentences, but the one used by Helgadóttir did not. Moreover, note that the results presented by Helgadóttir are based on all the ten test corpora.
[9] The following features in the POS tags were ignored: Case and gender for nominals, declension for adjectives, mood and voice for verbs, case for prepositions, and the type of proper nouns.
[10] One conceivable way of dealing with ambiguous case forms is to introduce ambiguous tags, e.g. one tag for words having the same form in accusative-dative-genitive.
[11] This seems logical since the missing tags were not found in the training corpus and are, hence, infrequent. Admittedly, this is a very simple smoothing strategy and experimenting with more sophisticated strategies would be worthwhile.
[12] The difference is statistically significant: $\alpha<0.025$, using McNemar's chi-squared goodness-of-fit test.
[13] In addition to using tag profile gap filling in the lexicon used by the TnT tagger, I performed an additional experiment for further improving the accuracy of TnT. The idea was to provide TnT with similar lexical information as used by

IceTagger for unknown words, i.e. supplying TnT with the tag profile for unknown words that were successfully analysed by the morphological/compound analyser of IceMorphy (recall that the tag profile for an unknown word analysed by the morphological/compound analyser is based on a related word found in the lexicon). For each such word, the corresponding tags were marked with the frequency 1 (i.e. in the absence of any frequency information, a uniform distribution was assumed). This slightly improved the overall tagging accuracy of TnT from 91.18% to 91.28%, and for unknown words from 72.75% to 74.31%. The reason for not a greater improvement is probably the fact the TnT already obtains relatively high accuracy for unknown words by applying only ending analysis based on a probability distribution. Contrastingly, IceTagger benefits significantly by using morphological/compound analysis, instead of using only ending analysis, as shown earlier.

[14] Since this paper was written, I have published a paper about different integration and combination methods for tagging Icelandic text; see Loftsson (2006b).