# Developing a PoS-tagged corpus using existing tools

**Hrafn Loftsson**[*], **Jökull H. Yngvason**[*], **Sigrún Helgadóttir**[†], **Eiríkur Rögnvaldsson**[‡]

[*]School of Computer Science, Reykjavik University, Iceland
[†]The Árni Magnússon Institute for Icelandic Studies, Iceland
[‡]Department of Icelandic, University of Iceland, Iceland
{hrafn,jokull06}@ru.is, {sigruhel,eirikur}@hi.is

### Abstract

In this paper, we describe the development of a new tagged corpus of Icelandic, consisting of about 1 million tokens. The goal is to use the corpus, among other things, as a new gold standard for training and testing PoS taggers. We describe the individual phases of the corpus construction, i.e. text selection and cleaning, sentence segmentation and tokenisation, PoS tagging with a combination method, error detection, and error correction. Furthermore, we discuss what problems have emerged, highlight which software tools have been found to be useful, and identify which tools are re-usable across different languages. Our preliminary evaluation results show that the error detection programs are effective and that our tagger combination method is crucial with regard to the amount of hand-correction that must be carried out in future work. We believe that our work will be of help to those wishing to develop similar resources for less-resourced languages.

## 1. Introduction

Language Technology (LT) for the Icelandic language has only existed for about a decade (Rögnvaldsson et al., 2009). In 2000, the Icelandic Frequency Dictionary (*IFD*) corpus (Pind et al., 1991) was the only LT resource. Since then, various resources have been developed, e.g. part-of-speech (PoS) taggers (Helgadóttir, 2005; Loftsson, 2008), a finite-state parser (Loftsson and Rögnvaldsson, 2007a), a lemmatizer (Ingason et al., 2008), and a morphological database (Bjarnadóttir, 2005). Icelandic is thus no longer a less-resourced language by any reasonable definition.

Before the work presented in this paper, the *IFD* corpus has been used to train and test PoS taggers (programs which automatically tag each word in running text with morphosyntactic information) on Icelandic text. The *IFD* corpus consists of about 590k tokens and all text fragments in the corpus were published for the first time in 1980–1989. In the tagset used, each character in a tag has a particular function. The first character denotes the word class and the remaining characters (up to 5) denote various morphological properties, for example, gender, number and case. The size of the *IFD* tagset is about 700 tags. The corpus was tagged with a special program that used grammatical rules and frequency information, derived from the manual tagging of 54k tokens (Briem, 1989). The automatic tagging was then hand-corrected line by line.

There are at least three problems associated with the use of the *IFD* corpus for developing PoS taggers. First, the corpus is relatively small in relation to the size of the tagset. Second, the underlying text has a strong literary bias. Third, the corpus text is 20–30 years old. Consequently, data-driven taggers, which have been trained on the corpus, may run into data sparseness problems, their accuracies may not be high enough when tagging different genres, and they may have difficulties when encountering recent linguistic phenomena which only occur in informal texts.

In this paper, we describe a work in progress, the development of a new tagged Icelandic corpus consisting of about 1 million tokens. It is intended that the corpus serve as a more suitable gold standard for developing PoS taggers than the *IFD* corpus. The new corpus will be tagged using the *IFD* tagset, although with some very minor modifications. The development of the corpus, henceforth referred to as the *GOLD*, consists of the following phases:

1. Text selection
2. Text cleaning
3. Sentence segmentation and tokenisation
4. PoS tagging
5. Error detection
6. Error correction
7. Evaluation

The emphasis in this work is on utilising existing tools and on automating the development process to as great an extent as possible. Phase 2 is semi-automatic and phases 3-5 are completely automatic. Our development platform is Ubuntu Linux and our software, which we intend to make open source, is written in shell scripts, Perl and Python.

In the remainder of this paper, we describe each of the above phases, discuss what problems have emerged, highlight which software tools have been found to be useful, and identify which tools are re-usable across different languages. Furthermore, we describe our preliminary evaluation results which show that the error detection programs are effective, and that our tagger combination method is crucial with regard to the amount of hand-corrections that inevitably must be carried out in order to make our *GOLD* a reliable standard.

## 2. Text selection

From 2004, work has been going on at the Árni Magnússon Institute for Icelandic Studies (AMI) to compile a tagged corpus of approximately 25 million tokens of texts (henceforth referred to as the *MIM* corpus) from different genres of contemporary Icelandic, i.e. texts written from the year 2000 and onwards.

Each word will be accompanied by a PoS tag and a lemma and each text will have bibliographic information attached. Emphasis has been placed on collecting written texts, both from printed sources and from texts found on the web. Just over 2% of the texts in the corpus will be transcribed spoken texts collected in connection with other projects (parliamentary speeches, conversations and interviews). Only texts that were available digitally were collected.

Permission has been sought from copyright owners of all the texts used. All copyright owners have signed a special declaration and agree that their material may be used free of licensing charges. In turn, AMI agrees that only 80% of each published texts are included and that copies of *MIM* are only made available under the terms of a standard licence agreement. The licence agreement will be modelled after the BNC User Licence[1].

*MIM* will be made available in two ways. Firstly, the corpus will be searchable on the website of the institute and, secondly, those that wish to use it in their own computers for language research or in LT can obtain a copy by signing the licence agreement. The corpus will be made available in TEI-conformant XML format (Burnard and Bauman, 2008).

The texts for the *GOLD* were selected from texts collected for *MIM*[2]. The texts were selected so as to reflect as far as possible the proportion of different types of text in *MIM*.

There are 13 different text types in the *GOLD*. In Table 1 these text types are listed together with the number of tokens of each type. As can be seen from the table this classification reflects to a great extent the origin of the texts. Texts were randomly sampled from textfiles from these different text types to the extent possible. None of the text samples in the *GOLD* is longer than 5,000 words. It should be noted that since permission has been granted by copyright owners of all the texts in the *MIM* the same applies to the text samples in the *GOLD*.

## 3. Text cleaning

The text samples for the *GOLD* were sampled after the *MIM* texts had been cleaned and prepared for tokenisation and tagging. In this section, we therefore give a short account of the process of cleaning the texts for *MIM*.

The texts obtained for *MIM* came in various formats. Text from most published books came as pdf-files. It is possible to extract text from pdf-files by using special programs: we mainly used a program developed by a member of our team. Some pdf-files are, however, rather difficult to handle and as a last resort we used optical character recognition software that is used for extracting text from scanned paper documents (ABBY FineReader: `http://finereader.abbyy.com/`). Some texts came in Word-documents which are easy to convert to text. Many texts were sampled directly from the Web so there was no need to change the format.

Text files varied greatly in quality. The cleanest text was obtained from the newspaper *Morgunblaðið*, taken directly from their database, classified by content. The text was sampled so as to reflect seasonal variation in topics under discussion. Their files, however, contained metadata that could be removed automatically. Some material was delivered as XML-files and we wrote a special program to extract clean text from those.

The text files obtained were either encoded using UTF-8 or ISO-8859-1 character encoding. It was decided that all texts in *MIM* should be converted to UTF-8 – hence all the texts in the *GOLD* use UTF-8 character encoding.

Texts from printed books and periodicals usually come with hyphenation. It was therefore necessary to run the texts through a program that joined the two parts of a word that had been split between lines. Various other measures had to be taken, either with automatic or semi-automatic means. We removed manually long quotations in a foreign language, long quotations from Old Icelandic texts and from new texts that we did not have permission to use, as well as footnotes, tables of content, indexes, reference lists, poems, tables and pictures.

Some texts were particularly difficult to handle and had to be fixed manually. This was particularly true for texts from the newspaper *Fréttablaðið* that were obtained as pdf-files. The text that was extracted from the files had to be rearranged to a certain extent.

As a final text cleaning step, we needed to carry out the following. Headings that do not end with an end-of-sentence marker (like a period, an exclamation mark or a question mark), but are only separated from the following text with a line-break, are common to many of the text types. For example:

```
Lokaorð
Bókmenntaverk verða að lögmálum ...
```

Here "Lokaorð" 'Epilogue' is the heading for the text below starting with "Bókmenntaverk".

The sentence segmentiser that we use (see the next section) does not consider such strings as being separate sentences, because in general a sentence can span multiple lines (with line-breaks in-between).

To handle this, we have written a program which searches for lines not ending with an end-of-sentence marker and whose following line starts with an upper-case letter. Such a pattern is a candidate for a heading followed by a new sentence. The program displays these occurrences to a user who then decides whether it is a correct candidate or not. If the candidate is correct, the program writes an additional line-break after the heading into the text file.

## 4. Sentence segmentation and tokenisation

Sentence segmentation and (word) tokenisation are often neglected, yet very important, pre-processing tasks. The former identifies where one sentence ends and another one

[1] `http://www.natcorp.ox.ac.uk/`. The crucial point in the licence agreement is that the Licencee can use his results freely but may not publish in print or electronic form or exploit commercially any extracts from the corpus other than those permitted under the fair dealings provision of copyright law.

[2] Note that, in contrast to *GOLD*, the tagging of *MIM* will not be manually checked and corrected.

begins, whereas the latter splits (for each sentence) a sequence of characters into meaningful linguistic units, like words, numbers and punctuation marks.

Developing a good sentence segmentiser/tokeniser is not a trivial task. For example, in the case of sentence segmentation, a period can serve as an end-of-sentence marker, as a decimal point, as a part of an abbreviation, etc. In the case of tokenisation, various things need to be accounted for, even when processing space-delimited languages. The most common of these is probably deciding when a punctuation character or other non-alphanumeric character should be part of the preceding character sequence or not. For example, a good tokeniser needs to be able to recognise occurrences of abbreviations, because otherwise they will be broken up into individual parts. Surprisingly little has been published regarding these important NLP tasks, but a good coverage of sentence segmentation and tokenisation can be found in (Grefenstette and Tapanainen, 1994; Palmer, 2000).

In our project, we rely on a sentence segmentiser and a tokeniser which are part of the *IceNLP* toolkit (Loftsson and Rögnvaldsson, 2007b). This toolkit is open source[3] and therefore we have been able to extend its functionality immediately when we have encountered new pre-processing errors (for example, by adding new abbreviations to the list of known abbreviations). Note that even though these tools were originally developed for processing Icelandic they should be applicable (or at least easily adjusted) to other related languages.

To a user, the segmentiser and the tokeniser is a single program which accepts an input file having a particular format and writes individual tokens to an output file in a given format. In our case, the input format is “free” (as opposed to one sentence per line) and the output format is one token per line with an empty line between sentences.

Obviously, the accuracy of the tokenisation is very dependent on the quality of the input. A typical first example occurs when the tokeniser splits a single word into two words, due to an erroneous additional space in the input. Consider the phrase “langan fangelsis dóm” ’long prison sentence’. In Icelandic “fangelsisdóm” is a single word, but since the tokeniser uses white space as a delimiter it has no chance of tokenising this correctly.

As a second example, note that a missing space (as opposed to an additional space) can also cause problems. Consider the string “c.Markmið” for which the correct string should have been “c. Markmið” ’c. Goal’. In this case, the tokeniser returns a single token for the string, because, generally, it allows a period to be a part of a token. However, a period is usually not a part of a string containing alpha characters unless the string is an abbreviation! Therefore, we have written a post-processing utility, which runs after the tokenisation, and fixes errors of this type (given a list of known abbreviations).

## 5. PoS tagging

Our PoS tagging phase consists of two parts. First, we tag the text with five individual taggers and then we apply a combination method to improve the tagging accuracy.

### 5.1. Individual taggers

The individual taggers that we use are (listed in descending order of accuracy when tagging Icelandic text): *IceTagger* (Loftsson, 2008; Loftsson et al., 2009), *Bidir* (Dredze and Wallenberg, 2008), *TnT* (Brants, 2000), *fnTBL* (Ngai and Florian, 2001), and *MXPOST* (Ratnaparkhi, 1996).

*IceTagger* is a linguistic rule-based tagger, specifically developed for tagging Icelandic text. It is a part of the *IceNLP* toolkit and thus open source. The other four taggers are data-driven, i.e. they learn a tagging model from a pre-tagged corpus. We obtained the bidirectional tagger *Bidir* from its developers, but *TnT*, *fnTBL* and *MXPOST* are downloadable from the web.

As discussed in Section 1, the *IFD* corpus has been used for training the data-driven taggers as well as developing *IceTagger*. The average tagging accuracy of the individual taggers used in the current project, measured using ten-fold cross-validation (and all the 700 tags in the tagset) against the *IFD* corpus, varies from around 89% to 92.5% (Helgadóttir, 2005; Loftsson, 2006; Loftsson et al., 2009).

All the taggers expect tokenised input. With the exception of *MXPOST*, the input format is one token per line with an empty line between sentences. *MXPOST* wants one sentence per line, and therefore we need to run a program which converts between these formats, both before and after tagging with *MXPOST*.

The *TnT* tagger is the only tagger that does not handle text in UTF-8 encoding. Thus, we needed to write a script which maps specific non-ASCII characters, like certain types of single quotes, to a character sequence that does not appear in the texts. For example, we map the single quotes ‘ and ’ to the characters strings BEGINSINGLEQ and ENDSINGLEQ, and then change the resulting file from UTF-8 encoding to ISO-8859-1 before *TnT* is run. When the tagger is finished we change the file back to UTF-8 and map the character strings back to the original quotes.

Once the tagging with the individual taggers has been carried out, we apply a few fixes to their output. For example, we make sure that the tag for every punctuation character is equivalent to the character itself, that number constants are always tagged with the same tag, and that abbreviations are always tagged in the same way.

### 5.2. Combined tagging

Finally, we apply a tagger combination method to the resulting output. Tagger combination methods are a means of correcting for the biases of individual taggers, and they are especially suitable when tagging a corpus, i.e. when effectiveness (accuracy) is more important than efficiency (running time). It has been shown that combining taggers will often result in a higher tagging accuracy than is achieved by individual taggers (Brill and Wu, 1998; van Halteren et al., 2001; Sjöbergh, 2003; Loftsson, 2006). The reason is that different taggers tend to produce different errors, and the differences can often be exploited to yield better results. We use *CombiTagger* (Henrich et al., 2009), an open source system[4], for carrying out the combination automatically. We feed the output of the individual taggers into *Combi-*

[3] http://icenlp.sourceforge.net

[4] http://combitagger.sourceforge.net

| Text type | Tokens | % of all tokens | Error candidates | True positives (%) | Tags corrected | % of tokens | Evaluation sample | Accuracy (%) |
|---|---|---|---|---|---|---|---|---|
| Newspaper 1[a] | 251,814 | 24.9 | 1,781 | 78.8 | 479 | 0.6 | 1,526 | 92.3 |
| Books | 237,204 | 23.5 | 1,663 | 77.7 | 1,438 | 0.6 | 1,247 | 95.1 |
| Blogs | 135,350 | 13.4 | 1,036 | 85.0 | 1,083 | 0.8 | 720 | 90.0 |
| Newspaper 2[b] | 94,749 | 9.4 | 750 | 79.2 | 724 | 0.8 | 1,021 | 87.6 |
| www.visindavefur.is[c] | 92,218 | 9.1 | 682 | 87.5 | 828 | 0.9 | 970 | 92.8 |
| Websites | 65,177 | 6.5 | 430 | 70.5 | 386 | 0.6 | 694 | 94.0 |
| Laws | 41,319 | 4.1 | 259 | 84.9 | 254 | 0.6 | 434 | 94.0 |
| School essays | 34,372 | 3.4 | 213 | 85.0 | 200 | 0.6 | 359 | 94.2 |
| Written-to-be-spoken | 19,348 | 1.9 | 142 | 85.2 | 151 | 0.8 | 202 | 92.1 |
| Adjudications | 12,880 | 1.3 | 101 | 96.0 | 148 | 1.1 | 134 | 88.1 |
| Radio news scripts[d] | 11,198 | 1.1 | 68 | 73.5 | 58 | 0.5 | 117 | 92.3 |
| Web media | 8,522 | 0.8 | 40 | 62.5 | 29 | 0.3 | 89 | 95.5 |
| E-mail | 5,513 | 0.5 | 54 | 88.9 | 59 | 1.1 | 58 | 89.7 |
| Total: | 1,009,664 | 100.0 | 7,200 | 80.9 | 5,837 | 0.7 | 7,571 | 92.3 |

Table 1: Information about the various text types in the new gold standard

[a] The newspaper *Morgunblaðið*. Only 500 error candidates out of 1,781 have been inspected, yet.

[b] The newspaper *Fréttablaðið*.

[c] A website operated by the University of Iceland where the public can post questions on any subject.

[d] The Icelandic National Broadcasting Service.

*Tagger* through the command line (a GUI interface is also available) and write the combined tagging result to a new file. The default combination method (which we indeed use) is simple voting (majority voting)[5], where each tagger gets an equal vote when voting for a tag and the tag with the highest number of votes is selected.

In *CombiTagger*, the resolution of ties depends on the exact order of the tagger files fed into the program. For example, if there is a voting tie between two tagger groups[6] $A$ and $B$ then the tag proposed by group $A$ is selected if one of its tagger's output has been loaded into *CombiTagger* before some output from group $B$. The order that we use is therefore descending order of accuracy as listed at the beginning of this section.

The whole process of tagging with the five taggers, applying fixes, and running *CombiTagger* is of course dependent on the size of the text being processed. To give an indication of the running time involved, it took 17 minutes (running on a Dell Precision M4300 2 Duo CPU, 2.20 GHz) processing the text type "Newspaper 2" which consists of 94,749 tokens (see Table 1), of which the *Bidir* tagger took close to 12 minutes and *CombiTagger* only 8 seconds. Thus, the whole tagging task processes about 93 tokens per second.

### 5.3. Discussion

We have not been able to find papers in the literature about corpus construction projects that follow our tagging method as described above, i.e. in which more than one tagger trained on the same corpus $C_1$ is used to tag another corpus $C_2$ (using the same tagset), followed by applying a tagger combination method. This is interesting because, as mentioned above, various researchers have demonstrated the effectiveness of applying combination tagging (Brill and Wu, 1998; van Halteren et al., 2001; Sjöbergh, 2003).

There may be several reasons for this. First, when one is confronted with the task of constructing the first tagged corpus for a language $L$, no data-driven tagger exists for $L$! Therefore, in order to apply a combination method for that task, one needs access to more than one tagger which is not data-driven, i.e. linguistic rule-based taggers, and, moreover, the taggers need to use the same tagset (if not, then a mapping between the tagsets needs to be possible). Linguistic rule-based taggers are, however, infrequent, let alone more than one such for a language $L$.

Second, if a corpus $C_1$ already exists for $L$, tagged with tagset $T_1$, then researchers may be reluctant to construct a new corpus $C_2$ for $L$, tagged with the same tagset. Indeed, projects have been carried out in which $C_2$ is tagged with a new tagset $T_2$ but, nevertheless, using taggers that have been trained on $C_1$. For example, in both (Zavrel and Daelemans, 2000) and (de Does and van der Voort van der Kleij, 2002), a stacking method was used to construct $C_2$, because $T_2$ was different from $T_1$ and/or the individual taggers used different tagsets. Stacking is a machine learning method which combines classifiers (taggers) by applying a classification algorithm using as features the tags chosen by the individual taggers. However, in this method, some amount of training data is still necessary, i.e. hand-annotated data that show which tag (from $T_2$) is correct for the combined classifier given the tags from the individual taggers.

## 6. Error detection

The output of the tagging phase is a single file consisting of tokens and the respective tags selected by *CombiTagger*.

[5] Other combination methods are possible, e.g. weighted voting or some user supplied voting algorithms.

[6] We use the term *tagger group* to denote a group of two or more taggers that agree on a particular tag.

Clearly, various tagging errors exist at this point, but, fortunately, some of the errors are systematic and can thus be detected automatically.

In a morphologically complex language like Icelandic, feature agreement, for example inside noun phrases, plays an important role. Therefore, of the total number of tagging errors existing in an Icelandic corpus, feature agreement errors are likely to be prevalent.

We use the noun phrase (NP), prepositional phrase (PP) and verb phrase (VP) error detection programs described by Loftsson (2009). In order to use these programs, a tagged corpus needs to be converted to one sentence per line and then parsed by *IceParser*, a finite-state parser which marks constituent structure and syntactic functions (Loftsson and Rögnvaldsson, 2007a)[7]. The output of the error detection programs is one error candidate per line. The parser and the error detection programs are not language independent. However, both are components of *IceNLP* and may thus be changed to work for other languages.

Let us consider two examples of error candidates. The first one is found by the noun phrase error detection program:

```
[NP [AP raunverulegt lhensf AP]
verðmæti nheo NP]
```

The above demonstrates a disagreement in case inside a noun phrase. The substring "[NP" denotes the beginning of a noun phrase, whereas "[AP" denotes the beginning of an adjective phrase (the AP is contained within the NP). The words in the phrase are "raunverulegt verðmæti" 'real value' and the corresponding PoS tags follow each word. The PoS tag "lhensf" denotes adjective (*l*), neuter (*h*), singular (*e*), nominative case (*n*), strong declension (*s*), and positive form (*f*). The PoS tag "nheo" denotes noun (*n*), neuter (*h*), singular (*e*) and accusative case (*o*). Thus, the noun is marked with accusative case but the adjective with nominative case.

The second example of an error candidate is found by the verb phrase error detection program:

```
{*SUBJ> [NP Ég fp1en NP] *SUBJ>}
[VP þekkti sfg3eþ VP]
```

This demonstrates a disagreement in person between the subject "Ég" 'I' and the main verb "þekkti" 'knew' (the substring "{*SUBJ>" denotes the beginning of the subject). The PoS tag "fp1en" denotes pronoun (*f*), personal (*p*), $1^{st}$ person (*1*), singular (*e*), nominative case (*n*). The PoS tag "sfg3eþ" denotes verb (*s*), indicative mood (*f*), active voice (*g*), $3^{rd}$ person (*3*), singular (*e*), past tense (*þ*). Thus, the subject is marked as $1^{st}$ person whereas the verb is marked as $3^{rd}$ person.

Both these error candidates signal a true error, but some of the candidates are *false positives* due to incorrect constituent marking by *IceParser*[8].

# 7. Error correction

We have written a program which inspects each error candidate and finds the line number in the tagged file where the first token associated with the error candidate occurs (in the first example above, the word "raunverulegt" occurs in line number 36,527 in the tagged file). The program outputs each error candidate along with the corresponding line number.

Once the error detection programs and the program for generating line numbers have been run, we load both the error candidates along with the line numbers and the tagged file into a spreadsheet. At that point, we start inspecting each error candidate, find its instance in the tagged file and correct the error if needed. Note that each error candidate can result in a correction of more than one tag.

Obviously, the *GOLD* contains errors other than the ones pointed to by the error detection programs. One frequent tagging error occurs in the tag for the first word of a sentence. The reason is that each sentence in the *IFD* corpus, the training corpus for the individual taggers, starts with a lower case letter (except in the case of proper nouns)! Therefore, all the taggers (except *IceTagger* which is not data-driven) very often tag a word at the beginning of a sentence with a proper noun tag. Indeed, we have written a program which points to likely errors at the beginning of a sentence, but we have not yet been able to inspect those candidates.

Finally, note that, before a corpus is published as a reliable gold standard, it has to be read, gradually line by line, and all tagging errors corrected. Clearly, the error correction that we have already carried out will speed up that process. For the line-by-line inspection, we intend to use some corpus correction software, e.g. *Posedit* which is open source[9]. In this final step, it is important to correct tokenisation errors as well.

# 8. Evaluation

In this section, we present two kinds of evaluations. First with regard to error detection and, second, regarding tagging accuracy.

## 8.1. Error detection

The total number of tokens in our *GOLD* is 1,009,664. Running on the whole corpus, the error detection programs output 7,200 error candidates, which is 0.7% of the total number of tokens. As previously stated, the development of the *GOLD* is a work still in progress. We have not yet finished inspecting all the error candidates, but at the time of writing we have inspected 5,919 of the 7,200 candidates (82.2%)[10]. This has resulted in 5,837 error corrections (corrections of PoS tags) in texts containing 837,334 tokens, i.e. we have had to correct 0.7% of the tokens based on the error candidates already inspected.

[7] When the PoS tags fed into *IceParser* are error free (hand-annotated), the F-measure of the parser for constituent structure is 96.7%. On the other hand, when PoS tags are produced by a tagger like *IceTagger* (thus containing some errors), the F-measure drops down to 91.9% (Loftsson and Rögnvaldsson, 2007a).

[8] Note that the set of false positives can be used to improve the parser!

[9] `http://elearning.unistrapg.it/corpora/posedit.html`

[10] For "Newspaper 1", we have only inspected 500 out of the 1,781 error candidates. As a result, we have made 479 corrections of tokens in texts containing 79,484 tokens, i.e. we have had to correct 0.6% of the tokens for "Newspaper 1" – see Table 1.

Information about the number of error candidates and the ratio of *true positives* for each text type can be seen in Table 1. The weighted average ratio of true positives is 80.9%. When applying the same error detection programs on the *IFD* corpus, Loftsson (2009) found 30.1% of the error candidates (448 out of 1489) to be true positives. The large difference can be explained by the fact that the *IFD* corpus has been corrected line by line whereas our *GOLD* has not – yet.

### 8.2. Tagging accuracy

In order to estimate the tagging accuracy of the individual text types, we performed the following. For each text type, we sampled every $100^{th}$ word (for "Newspaper 1", "Books" and "Blogs" we have only finished sampling 50-60% of the texts), i.e. 1% of the corresponding text. For each sampled word, we manually checked whether its tag was correct or not. A tag is correct if the whole tagstring (consisting of up to 6 letters) is correct.

The results can be found in the last two columns in Table 1. The 95% confidence limits for the estimated accuracy are acceptable for the largest samples (400 tokens or more, e.g. for Websites: ±1.92%). The smallest samples (sample size less than 200) are, however, too small and need to be enlarged to give more reliable results. However, the resulting tagging accuracy achieved is very encouraging. For many of the text types, "Books", "Websites", "Laws" and "School essays", the tagging accuracy is $\geq$ 94%. Note that these texts contain continuous texts of good quality.

For "Books" the accuracy is above 95% which seems very good compared to the best tagging result of 93.5% using the *IFD* corpus (whose text is of similar type as our "Books") obtained by Loftsson (2006) when applying a simple voting method using five taggers. The main difference between our combination and the one used by Loftsson is twofold. First, we extend the dictionaries of *IceTagger* and *TnT* with part of the data from the Morphological Database of Icelandic Inflections (MDII) (Bjarnadóttir, 2005)[11]. By using data from the MDII, the ratio of unknown words in *IceTagger* and *TnT* is significantly lower than in the other three taggers. Note that due to the different unknown word ratio, we do not present tagging accuracy for unknown words and known words separately in Table 1. Second, we use the *Bidir* tagger in the combination instead of the *MBT* tagger (Daelemans et al., 1996) – the former is significantly more accurate than the latter when tagging Icelandic text.

The accuracy of four of the text types is $\leq$ 90%, i.e. "Blogs", "Newspaper 2", "Adjudications" and"E-mail"). For e-mails and blogs, this is not surprising, because the structure of sentences in these texts is sometimes unconventional and usually informal, and they often contain high frequency of foreign words and unconventional spelling. The relatively low accuracy of the adjudications texts can be explained by the fact the the word order is often "stilted", which the underlying tagging models sometimes have difficulties with.

For the "Newspaper 2" texts (*Fréttablaðið*), the taggers often have difficulties with foreign words (e.g. proper nouns), abbreviations, headlines, etc., and, moreover, these texts contain classified ads which can be difficult to tag for the individual taggers. Furthermore, as mentioned in Section 3, these texts were particularly difficult to handle and had to be fixed manually.

The tagging accuracy for the "Newspaper 1" text is much better compared to "Newspaper 2". As mentioned in Section 3, the "Newspaper 1" text was taken directly from the database of the publisher, classified by content and was therefore relatively clean. No classified ads were contained in the text. It is therefore not surprising that the tagging accuracy is better than for newspaper text that had to be extracted from pdf-files.

#### 8.2.1. Error examples

The tagging errors found during our estimation of tagging accuracy are of various kinds. Most of them do not seem to be systematic, and hence we have not been able to write programs to correct them automatically. Below we give three examples of output from *CombiTagger* showing different kinds of errors found in the text type "Books". The first two columns show the word and the tag, respectively; in the third column we show an English gloss.

First, consider the sentence fragment:

| | | |
|---|---|---|
| það | fphen | it |
| virðist | sfm3en | seems |
| falla | sng | fit |
| vel | aa | well |
| að | c | to |
| staðalmynd | nven | stereotype-the |
| samfélagsins | nheeg | society's-the |

This fragment contains two errors because "að staðalmynd" should be tagged as "að *aþ* staðalmynd *nveþ*", i.e. "að" as a preposition governing the dative case (instead of a conjunction (*c*)), and "staðalmynd" as a noun (*n*), feminine (*v*), singular (*e*) and dative case (*þ*) (instead of nominative case (*n*)). Only one of the five taggers, the *fnTBL* tagger, tags these two words correctly.

The second example demonstrates a long-distance dependency which is often difficult for taggers to handle correctly:

| | | |
|---|---|---|
| þær | fpvfn | they |
| fóru | sfg3fþ | went |
| að | cn | to |
| mennta | sng | educate |
| sig | fpkeo | themselves |

Here "þær" is correctly tagged as a pronoun (*f*), personal (*p*), feminine (*v*), plural (*f*), nominative case (*n*), but the reflexive pronoun "sig", is incorrectly marked as masculine (denoted by the third letter (*k*) in the tag) and singular (denoted by the fourth letter (*e*)) instead of feminine and plural, because it refers to the word "þær". Only one of the five taggers, *IceTagger* in this case, tags the word "sig" correctly.

The last example demonstrates an incorrectly tagged word-class:

| | | |
|---|---|---|
| Sá | faken | that |

[11] This database is accessible from http://bin.arnastofnun.is/

| | | |
|---|---|---|
| æsti | sfg3eþ | upset (man) |
| verður | sfg3en | becomes |
| stöðugt | aa | consistently |
| ágengari | lvenvm | (more) agressive |

Here “æsti” is tagged as a verb (the first letter in the tag *sfg3eþ* denotes a verb) but should be tagged as an adjective. None of the taggers tags this word correctly because this particular word form does not exist as an adjective in the dictionaries used by the taggers – it only exists as a verb.

## 9. Conclusion

In this paper, we have described the development of a new corpus, *GOLD*, of Icelandic text. *GOLD* consists of about 1 million tokens and will be used as a gold standard for training and testing PoS taggers. We have described the individual phases of the corpus development, text selection and text cleaning, sentence segmentation and tokenisation, PoS tagging, error detection and error correction, and, finally, evaluation results.

We have identified which tools have been of help during the development and which tools are usable across different languages. We believe that our work will be of help to researchers wishing to develop similar resources for less-resourced languages.

Our evaluation of tagging accuracy indicates that the error detection programs are effective and that the extra effort of applying five taggers and a combination method is crucial with regard to the amount of hand-correction that inevitably must be made in order to use *GOLD* as a reliable gold standard in the future.

Finally, since the methods applied in the construction of *GOLD* have been successful, we intend to use the same methods when tagging *MIM*, the corpus of 25 million tokens of modern Icelandic texts. The only difference is that we do not foresee a line-by-line inspection of the tagging for *MIM*!

## Acknowledgments

The work in this paper was partly supported by both the Icelandic Student Innovation Fund and the Icelandic Research Fund, grant 090662012.

## 10. References

K. Bjarnadóttir. 2005. Modern Icelandic Inflections. In H. Holmboe, editor, *Nordisk Sprogteknologi 2005*. Museum Tusculanums Forlag, Copenhagen.

T. Brants. 2000. TnT: A statistical part-of-speech tagger. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA, USA.

S. Briem. 1989. Automatisk morfologisk analyse af islandsk tekst. In *Papers from the Seventh Scandinavian Conference of Computational Linguistics*, Reykjavik, Iceland.

E. Brill and J. Wu. 1998. Classifier Combination for Improved Lexical Disambiguation. In *COLING-ACL ’98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, Montreal, Quebec, Canada.

L. Burnard and S. Bauman. 2008. Guidelines for Electronic Text Encoding and Interchange P5 edition. Text Encoding Initiative. http://www.tei-c.org/Guidelines/P5/.

W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. 1996. MBT: a Memory-Based Part of Speech Tagger-Generator. In *Proceedings of the 4th Workshop on Very Large Corpora*, Copenhagen, Denmark.

J. de Does and J. van der Voort van der Kleij. 2002. Tagging the Dutch PAROLE Corpus. In M. Theune, A. Nijholt, and H. Hondorp, editors, *Language and Computers – Computational Linguistics in the Netherlands 2001. Selected Papers from the Twelfth CLIN Meeting*. Rodopi, Amsterdam-New York.

M. Dredze and J. Wallenberg. 2008. Icelandic Data Driven Part of Speech Tagging. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Columbus, OH, USA.

G. Grefenstette and P. Tapanainen. 1994. What is a word, What is a sentence? Problems of Tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, Budapest, Hungary.

S. Helgadóttir. 2005. Testing Data-Driven Learning Algorithms for PoS Tagging of Icelandic. In H. Holmboe, editor, *Nordisk Sprogteknologi 2004*. Museum Tusculanums Forlag, Copenhagen.

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

A. K. Ingason, S. Helgadóttir, H. Loftsson, and E. Rögnvaldsson. 2008. A Mixed Method Lemmatization Algorithm Using Hierachy of Linguistic Identities (HOLI). In B. Nordström and A. Ranta, editors, *Advances in Natural Language Processing, 6th International Conference on NLP, GoTAL 2008, Proceedings*. Gothenburg, Sweden.

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

H. Loftsson and E. Rögnvaldsson. 2007b. 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.

H. Loftsson, I. Kramarczyk, S. Helgadóttir, and E. Rögnvaldsson. 2009. Improving the PoS tagging accuracy of Icelandic text. In *Proceedings of the 17th Nordic Conference of Computational Linguistics (NODALIDA-2009)*, Odense, Denmark.

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

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

H. Loftsson. 2009. Correcting a PoS-tagged corpus using three complementary methods. In *Proceedings of the* $12^{th}$ *Conference of the European Chapter of the ACL (EACL 2009)*, Athens, Greece.

G. Ngai and R. Florian. 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.

D. Palmer. 2000. Tokenisation and Sentence Segmentation. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*. Marcel Dekker, New York.

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

A. Ratnaparkhi. 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.

E. Rögnvaldsson, H. Loftsson, K. Bjarnadóttir, S. Helgadóttir, A. B. Nikulásdóttir, M. Whelpton, and A. K. Ingason. 2009. Icelandic Language Resources and Technology: Status and Prospects. In R. Domeij, K. Koskenniemi, S. Krauwer, B. Maegaard, E. Rögnvaldsson, and K. de Smedt, editors, *Proceedings of the NODALIDA 2009 Workshop Nordic Perspectives on the CLARIN Infrastructure of Language Resources*. Odense, Denmark.

J. Sjöbergh. 2003. Combining POS-taggers for improved accuracy on Swedish text. In *Proceedings of NoDaLiDa 2003*, Reykjavik, Iceland.

H. van Halteren, J. Zavrel, and W. Daelemans. 2001. Improving Accuracy in Wordclass Tagging through Combination of Machine Learning Systems. *Computational Linguistics*, 27(2):199–230.

J. Zavrel and W. Daelemans. 2000. Bootstrapping a Tagged Corpus through Combination of Existing Heterogeneous Taggers. In *Proceedings of the* $2^{nd}$ *International Conference on Language Resources and Evaluation (LREC 2000)*, Athens, Greece.