Continued development of Apertium-IceNLP: A rule-based Icelandic to English machine translation system

Ólafur Waage
School of Computer Science
Reykjavík University
IS-101 Reykjavík, Iceland
olafurw09@ru.is

Abstract

We describe the continued development of the open source rule based Icelandic to English machine translation system (Brandt et al., 2010). It was created by combining the Apertium machine translation framework and adding the IceNLP toolkit with the purpose to increase the translation accuracy. It had an WER of 50.6% and a PER of 40.8%, scoring worse than the pure Apertium is-en language pair it was based on.

We removed IceMorphy from the translation pipeline and replaced it with Apertium's morphological analyzer (lt-proc). Then, we created a parser and mapper from Apertium's stream format to IceNLP's tag format for IceTagger to use. After this was completed, and after fixing the errors that cropped up along the way, the updated WER is 40.5% and PER is 21.3%.

We conclude that increased accuracy might be achieved by looking deeper into Icelandic multiword expressions.

1 Introduction

Machine translation (MT) has come a long way in the last few years but accurately translating Icelandic has been a demanding task. A step forward was made in 2010 when a prototype machine translation system was created by combining Apertium and IceNLP (Brandt et al., 2010). It was achieved by adding IceMorphy and IceTagger as the first two parts of the translation system, replacing similar parts from the Apertium framework. The question it tried to answer was: Would the translation quality of Apertium increase when adding IceNLP modules to the Apertium translation pipeline?

The results of that prototype were that it had a word error rate (WER) of 50.6% and an position-independent word error rate (PER) of 40.8%, scoring worse than the pure Apertium translation pipeline. Both Apertium and Apertium-IceNLP also scored worse than other Icelandic to English (is-en) translation system (Tungutorg and Google Translate). The conclusion was that using only IceTagger and removing IceMorphy from the pipeline might increase the accuracy, since IceMorphy has a poor support for multiword expressions, which Apertium handles well.

The work described in this paper goes through the process of removing IceMorphy from the existing Apertium-IceNLP prototype and replacing it with Apertium's morphological analyzer (lt-proc). We describe the creation of a parser and a mapper from the Apertium tagset format to the IceNLP tagset to be used by IceTagger and the errors fixed along the way. Furthermore, we describe errors like the differences between the tagsets, incomplete and wrong tag databases and missing frequency information required for IceTagger to function properly.

Results of this continued development of the Apertium-IceNLP prototype were positive. Apertium-IceNLP's WER and PER scores are 40.5% and 21.3%, respectively, while the score of pure Apertium is 42.8% (WER) and 23.8% (PER). These scores cannot be directly compared with the previous results since the testing was done using different test data, though they can give an estimate of the improvement done.

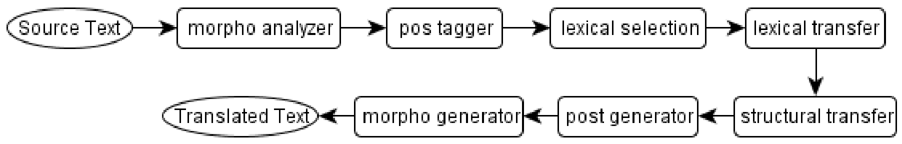

Figure 1: The Apertium Pipeline

2 Background

Translating Icelandic using a computer is a demanding task that requires both in-depth knowledge of the language and highly technical software. Over the years the accuracy of Icelandic translation has increased and, in 2010, Martha Dís Brandt created a prototype machine translation system for her Master's thesis, where she combined Apertium (an existing shallow-transfer machine translation system) and IceNLP (an Icelandic NLP toolkit). The result of that prototype was not a positive one but there were ideas on how to improve the system.

2.1 Apertium

Apertium (Forcada et al., 2009), was developed originally to translate from Spanish to Catalan, and Spanish to Portuguese; designed in such a way that adding new language pairs was simple. It is a shallow-transfer MT system, which means that the system does not perform deep analysis on the source text. The Apertium system is setup as a pipeline of various programs that function independently from each other, so replacing them is relatively simple (Forcada et al., 2009).

Each language pair is a work of the Apertium community and anyone can join and start a new language pair or assist in improving an existing one. The Icelandic to English language pair has a 45.9% word error rate (WER) and a 38.2% position-independent word error rate (PER) (Brandt et al., 2010)

2.2 IceNLP

IceNLP is a natural language processing (NLP) toolkit used to process and tag Icelandic text (Loftsson and Rögnvaldsson, 2007b). It contains a tokenizer and a morphological analyser (IceMorphy), a lemmatizer (Lemmald; (Ingason et al., 2008)), a linguistic rule-based part-of-speech (POS) tagger (IceTagger; (Loftsson, 2008)), a trigram tagger (TriTagger), and a shallow (finite-state) parser (IceParser; (Loftsson and Rögnvaldsson, 2007a)).

The IceNLP tools focused on in this paper are IceMorphy and IceTagger. IceMorphy morphologically analyses text to find all possible tags from a given word. It does not look at an entire sentence but only one word at a time. IceTagger then takes this list of tags for each word and selects the most likely tag for the word based on the sentence.

The sentence `Ég er góður.` would be analyzed and tagged by IceNLP like this `Ég fp1en er sfg1en góður lkensf . .`

The tags in between the words mark the tag of the preceding word. `fp1en` describes a personal pronoun, first person, singular and nominative, which fits for the word `Ég` in this case.

2.3 Apertium-IceNLP

A prototype of combining these two systems was started by Brant for her Masters thesis. The system utilizes the existing pipeline within Apertium to translate Icelandic text analyzed by IceMorphy/Lemmald and tagged by IceTagger. It had an error rate higher than other similar systems, scoring worse than the pure Apertium pipeline.

The hypothesis was that the IceMorphy/Lemmald analysis was not good enough so the next task was to replace it with the Apertium analyzer.

The pipeline (as seen in figure 1) starts with a morphological analyser (lt-proc) and its results are sent to the tagger to select the correct tag for each word. In the original prototype, the idea was to swap out lt-proc and the Apertium tagger for IceMorphy/Lemmald and IceTagger. IceTagger and IceMorphy were chosen since they start the tagging process and IceTagger has a high tagging accuracy for Icelandic text.

A problem with combining IceNLP and Apertium is that they use different tagsets. IceNLP uses a single letter for each morphological feature and does not have any other syntax attached to it. For example the tag `nkfng` would describe a noun, masculine, plural, nominative and definitive

Translator	WER	PER
Apertium-IceNLP	50.6%	40.8%
Apertium	45.9%	38.2%
Tungutorg	44.4%	33.7%
Google Translate	36.5%	28.7%

Table 1: Testing results

respectively. Within Apertium this same tag would be `<n><m><pl><nom><def>`. This problem was solved by using a tagmapping file that had an IceNLP tag on the left and the corresponding Apertium tag on the right.

The final setup would then utilize IceMorphy to analyze the input text, send that result to IceTagger which would tag the text and ouput an Apertium formatted result which would continue down the rest of the Apertium pipeline, resulting in a translated text.

2.3.1 Original Apertium-IceNLP Results

To produce WER and PER scores, a tool called `apertium-eval-translator` was used. This tool has been used by Apertium before to calculate scores of their language pairs so it was chosen to be able to compare the results to other Apertium systems.

One thousand sentences were randomly chosen and then filtered to produce 397 usable ones. This size was chosen to give a reasonable number of sentences after filtering. The reason for filtering was due to the randomness of the selection and it's source, since many of these sentences do not aid in giving a good estimation of the systems quality. For example, sentences need to have three or more words, have less than two unknown words, are not tabular, etc.

These sentences were translated using Apertium, Apertium-IceNLP, Tungutorg and Google Translate. Then those sentences were copied and minimal corrections where made to make them as grammatically correct as possible.

These files were then processed with `apertium-eval-translator` giving the scores shown in Table 1.

To decrease the error rate, it might be sufficient to simply replace the POS tagger of Apertium and leave out IceMorphy and Lemmald. The reason is that IceMorphy and IceTagger do not handle multiword expressions (MWE) at all, and IceTagger wants to tag them as individual words. Apertium handles MWE's very well and this would most likely give desired results, since Icelandic uses a lot of phrases that cannot be translated directly.

There are many examples of Icelandic MWE's that are only one word in English. Examples of this include:

```
Af hverju -> Why
Vegna þess að -> Because
Á meðan -> While
```

3 Development

The aim of the development covered in this report focused on removing IceMorphy and Lemmald from the Apertium-IceNLP pipeline and leave only IceTagger behind. This would require writing a parser and a mapper from the Apertium tagset to IceNLP the tagset so that minimal changes would be needed to IceTagger. No further change would be required, since IceTagger could already output Apertium style tags. This work was therefore only focused on the first part of the pipeline process.

3.1 Changes to IceTagger

Even though IceTagger would theoretically not need any changes, some were made to simplify the implementation of the new changes.

A new flag was added to the IceNLP toolkit. `-x <icenlp|apertium>` which either uses IceMorphy or lt-proc as the morpho analyzer respectively. If that flag is set to Apertium, then the mapping process is the first one to run within Apertium-IceNLP. Also a slightly different tagging function is called from IceTagger since dictionary token lookup is only done to unknown words (to help IceTagger tag the surrounding words correctly) and no dictionary lookup is needed to find the lemma since lt-proc provides it.

3.2 Switching out IceMorphy

The previous version of Apertium-IceNLP got the raw source language sentence and was able to parse it in its natural way. But by using lt-proc as the entry point, the input string becomes a formatted text stream so it needs to be modified and parsed before IceTagger is able to do its work. The pipeline was setup as a command line script where the output of lt-proc was piped into the IceNLP executable.

There, a system of parsing tools take the stream; process and prepare it so little modification is needed on the IceTagger side.

Following are simple explanations of each part of the mapping and parsing tools created.

3.2.1 ApertiumSegmentizer

This class is the first one to be run on the text from lt-proc. It goes through the text stream to find the end of a sentence. It searches for `<sent>$`. This is a tag used by Apertium to mark the end of sentences and is used as a tag after periods and question marks. It uses a simple StringBuilder and adds one character at a time. When it is adding a `$` to the string, it checks if the last 7 characters are the end of sentence tag, and if so, the reading is stopped and the string collected is returned.

This class can be used in a loop to check if there are more sentences and the class is not specific to IceNLP.

3.2.2 LtProcParser

This class receives the sentence from the segmentizer and is responsible for splitting the sentence into words. The Apertium stream format uses ^ as a start of a word and ends it with a `$`. So a simple regular expression is used to split the string up `^|^ $|^$|$`

```
^Glaður/glaður<adj><pst><m><sg>
<nom><sta>$
```

When split, each sentence (string) consists of a starting symbol ^ and an end symbol `$`. After the start symbol comes the lexeme of the word in question. After the lexeme comes a list of all possible tags, starting with a forward slash, then the lemma and finally the tags. Words that are unknown have the lexeme unchanged and one lemma with an asterix in front of it with no tags eg. `dýralíf/*dýralíf`.

To store each word, an ApertiumEntry class was created. One Apertium entry is one word and can contain zero or more LexicalUnit objects. It also stores the surface form of the word. The LexicalUnit class contains the lemma, tags and other properties that can assist in parsing the data.

LtProcParser needs to handle edge cases as well, such as the invariable part of a multiword marker, where only the first word of a multi word expression is tagged and the rest of the expression is attached to the end of the lexical unit.

An example of an invariable multiword marker is the multiword expression `var í samræmi`

```
^var í samræmi/vera<vblex><past>
<p1><sg># í samræmi/vera<vblex>
<past><p3><sg># í samræmi$
```

Here the only lemma considered is `vera` and the rest of the expression is tagged along behind.

After parsing the sentence, LtProcParser returns an array of ApertiumEntry objects to be used by the IceNLPTokenConverter class.

3.2.3 IceNLPTokenConverter

Here the ApertiumEntry objects are processed and converted into IceNLP tags. IceNLPTokenConverter relies heavily on the tag profile (the list of possible tags) of each word and the tag map used by IceTagger to convert IceNLP tags to Apertium tags.

The process that IceNLPTokenConverter goes through is to detect the first tag of each LexicalUnit within the ApertiumEntry object. Then different methods are used to select an IceNLP tag. There are methods for unknown words, prepositions, verbs and then there are methods that convert tags directly.

But before this is done, the lexical units of each Apertium entry is cleaned. This process changes such tags as `<np><org>` to `<np><al>` since IceNLP does not differentiate between proper nouns with the same detail as Apertium.

The normal mapping process is to find the word in a tag dictionary. These dictionaries are a map of words and their tags sorted by usage frequency. When a word is found in one of these dictionaries, the tagset there is used and the ltproc tags are ignored. The tags used in the mapping process are based on the Frequency Dictionary of Icelandic (Pind et al., 1991).

3.2.4 Frequency information

A problem with the Apertium is-en language pair is that the tag profile returned by lt-proc is not sorted in frequency order. This is a disadvantage for IceTagger because if there are more than one possible tags after disambiguation then IceTagger picks the first one, assuming it is the most frequent one.

This information is retained when using IceMorphy and IceTagger, but when replacing IceMorphy with lt-proc, then it is lost.

This is stored within IceNLP in two files (baseDict.dict and otb.dict), an example of an entry within baseDict.dict is:

```
eigi=svg3en_svg1en_svg3fn_aa
```

It has the word, and then an equal sign followed by the tag profile. These values are sorted by frequency, so the most common tag is the first one.

Without this information, the whole process would perform worse than before, so some method must be derived to use the lt-proc tags but sort them based on the frequency information in these files.

IceNLPTokenConverter does these conversions in the following manner.

For normal mapping (not a verb, preposition, multiword expression or an unknown word) the frequency dictionaries are searched for an entry using the lexeme of the word. If those entries are found, then the first tag within that entry is converted into an Apertium style tag and that tag is searched for within the lt-proc results. If found, it is added to the list to be returned and the next tag is searched for in the same fashion. If it is not found, then it continues with the next tag.

When it is done with all the tags from the dictionary and there are more tags to be added from lt-proc, they are added to the list in the same order they are currently in.

If IceNLPTokenConverter is dealing with a verb, then the exact same method is applied but it trusts the frequency dictionary tag list and only adds to the list tags from lt-proc that appears in the tagset list from the dictionary.

3.3 Differences between tagsets

When a running version of the mapping tools was up, a few tests were done to confirm that everything would be running properly. A majority of the problems with switching out IceMorphy for lt-proc was not the parsing or programming, but the differences between the Apertium and IceNLP tagsets. The prior development had created an IceNLP to Apertium mapping and a decision was made to use that same mapping since at the time it seemed like it would work with minimal problems.

The mapping file has the IceNLP tags, whitespace then the corresponding Apertium tag. Work had been done to sort out all errors for mapping from IceNLP to Apertium but using those same mappings backwards would account for a lot of work.

3.3.1 One to One

There are a few different tags in IceNLP that describe a preposition but within Apertium there is only one tag.

```
aþm <pr>
aþe <pr>
ao <pr>
aþ <pr>
ae <pr>
```

This is fine when mapping from IceNLP to Apertium but when going back you will have problems, so when reading into the reverse Hash Map the last tag in the file with the string `<pr>` is used, for example, the tag `ae` above. A solution to that is to have the most important tag the last one. The Apertium tagset does not have the information to differenciate between the tags.

3.3.2 Missing tags

The tag map included all the necessary IceNLP tags but it did not capture all of the Apertium tags. An early error was discovered when the parsing process did not find a tag, it was left empty and IceTagger then started guessing what tags to use. This was a difficult error to spot at first since the output looked pretty normal, especially if IceTagger guessed correctly.

A large text corpus was tagged frequently to keep track of how many of these errors were left. They were looked at individually and fixed one by one, since sometimes the error was only a missing entry in the mapping file but other times it was an error in the Apertium is-en language pair.

3.3.3 Combined tags

Another early error was that Apertium used combined tags for words that could be both male/female and singular/plural. Usually when a word could be both male and female in the same form, two different entries would be created for that word within Apertium. One with the `<m>` tag and one with the `<f>` tag, but sometimes a combined tag would be used `<mf>`. An example of this was the word `fleiri` (more). Here is an example sentence and output from lt-proc early in the development process. (formatting mine):

```
Það voru fleiri strákar í hópnum
There were more boys in the group.
```

```
^fleiri
/fleiri<det><qnt><mf><sp><nom><sta>
/fleiri<det><qnt><mf><sp><acc><sta>
/fleiri<det><qnt><mf><sp><dat><sta>
/fleiri<det><qnt><mf><sp><gen><sta>
$
```

Here the word `fleiri` is tagged as a determiner, quantifier, male/female, singular/plural, nominative/accusative/dative/genative and with a strong inflection. Since IceNLP does not contain a tag that describes a word belonging to both genders and both tenses the fix was to increase the data sets from the current 4 to 20. This would include 12 lines for male, female and neuter in plural and 8 lines for male and female in singular, since neuter singular would be `fleira`.

3.3.4 Strong vs Weak Inflection

As seen in the example above, `fleiri` is marked as having a strong inflection. In Icelandic, this is not correct, as the strong inflection would have the root `flest (eg. flestir, flestar)`. This error is based on the fact that the word is classified as a determiner. This is also incorrect, since `fleiri` is an adjective.

There were also cases where the `<sta>` `<vei>` tags were not within the tag mapping from IceNLP to Apertium. In some cases those tags were not used in determining the correct tag to use, causing their removal from the Apertium entry list, for example on past participle verbs like `ætluðu`. But in instances where the tag was required, it was added to the mapping list.

3.3.5 Missing lemmas

A large part of the error hunting was to hunt down missing lemmas. Since the lemma information used is from lt-proc, if the data is not mapped correctly then IceTagger starts guessing what tags the word should have been based on its neighbors. But not using IceMorphy means that Lemmatizer cannot be used to find the lemma of a word.

To solve this problem, one would either have to go through each word where this would happen, find the source of the bug and fix it, or try to find the lemma within the original parsed data from lt-proc.

The second method was explored a bit before finding the flaw in that approach. What if lt-proc does not have the correct lemma for the tag proposed by IceTagger (e.g. IceTagger says that a word is an adjective but the lemmas that lt-proc has are only for verbs)? A simple lemma guesser was written and used for a while but then removed when the number of actual errors went down.

Fixing the missing lemma problem was difficult; not due to the complexity of each individual problem; but due to the number of different problems. Some were easily fixed by adding the missing tags in either the mapping file or within the is-en language pair in Apertium. But others required more extensive fixes.

An example of this was the word `sig`. This problem started out as a missing lemma problem but then branched out into a larger translation problem, since translating the word `sig` is dependent on the gender.

```
Hann meiddi sig -> He hurt himself
Hún meiddi sig -> She hurt herself
Það meiddi sig -> It hurt itself
```

This was not handled correctly and took a considerable amount of effort to get right, since changes were needed within Apertium is-en language pair.

3.4 IceNLPServer

An Apertium-IceNLP webservice had been created to be used as a front end to the translation service (Brandt et al., 2010). Since it used the Apertium-IceNLP tool in a different way, some changes were needed there.

Due to the quality of the code design of the server, very little change was needed. But since the mapping process now receives the data first, then sends that data to IceTagger and that feature request was not predicted when the server was created, a few places needed change.

Since IceTagger is using a different tagging function as mentioned above, the server needs to be setup to handle the same configuration flag and it needs to be added to the servers startup workflow.

4 Evaluation

To be able to compare these results with the previous results done by Brandt, the same evaluation method was used. It consisted of randomly selecting 1000 sentences from a corpus and then filtering them. For example sentences need to have three or more words, have less than two unknown words, are not tabular data etc. That filtering process resulted in 350 sentences. Those sentences where then translated with the pure Apertium is-en pair and also with Apertium-IceNLP.

The test corpus came from the Icelandic newspaper Morgunblaðið. Roughly 1.5 million sentences were used in the random selection process.

Minimal corrections where made to the sentences to make them as grammatically correct as possible. The difference between the fixed version and the original translated version is the error percentage shown in Table 2. There are two types of error percenteges. Word error rate (WER) where

Translator	WER	PER
Apertium	42.8%	23.8%
Apertium-IceNLP	40.5%	21.3%

Table 2: Testing results

the word in that particular spot is wrong and a position-independent word error rate (PER) where the error is only if a word isn't supposed to be in the sentence, so the position of a correct word does not matter.

The tool used to calculate the errors is called `apertium-eval-translator` and is a standard tool used within the Apertium communuty to calculate these two error rates.

Only the differences between each translator set can be directly compared. Our numbers are not directly comparable to the ones obtained by Brandt, because we use different test data and the corrections are carried out by different people.

Here we see that the Apertium-IceNLP has a WER of 2.2 percentage points lower and a PER of 2.5 percentage points lower than pure Apertium.

This shows that using IceTagger within the Apertium translation system does increase the translation accuracy from Icelandic to English.

5 Discussion and future work

Icelandic is morphologically very complex and translating sentences directly might work in some cases but most often the meaning is lost. Adding IceTagger has increased the accuracy of the translation but would an Icelandic tagger with 100% accuracy be enough to give the best translation? I would think not, since many of the translation errors are due to the multitude of Icelandic multiword expressions. These expressions need to be collected and mapped to increase the accuracy significantly.

Let's look at an example from the testing sentences.

```
Original > Hægt er að bera saman.
Translation > Slow is carrying together.
Fixed Translation > You can compare.
```

This is a great example of direct translation losing the meaning of a sentence. Phrases like this one make translating Icelandic tough since these errors cannot be fixed by higher tagging accuracy or a larger tagged corpus. They might be fixed by training the system with these kinds of phrases, but that might only fix that particular phrase or you might risk overfitting.

Icelandic is often about using the right word in the right sentence, or else the whole sentence can lose its meaning or be misunderstood. Fixing this correctly might not be feasible but getting closer might be by looking at what is causing those errors and categorizing them by the type of the error.

6 Conclusion

In this article, I have described the Apertium-IceNLP Icelandic to English translator and the translation accuracy it displays and the work that has been done to increase that accuracy by removing IceMorphy due to its lack of handling multiword expressions.

An Apertium stream parser and mapper to the IceNLP tag format was created to allow use of the Apertium morphological analyser as a first part of the translation process. In the first article by Brandt the translation pipeline started of with IceMorphy and then continued over to IceTagger, but here lt-proc (Apertium's morpho analyzer) was used as an entry point.

I discussed the problems in mixing tagsets, the information loss and the general timesink in fixing related problems. Problems like combined tags in the Apertium tagset and IceTagger relying on tag frequency sorting when lt-proc returns the tags in an unsorted manner.

The result of this continued development of Apertium-IceNLP show that it has a 2.2 percentage points lower word error rate and a 2.5 percentage points lower position independent word error rate than the pure Apertium is-en language pair.

References

Brandt, Martha Dís, Hrafn Loftsson, Hlynur Sigurþórsson, and Francis M. Tyers. 2010. Apertium-IceNLP: A rule-based Icelandic to English machine translation system. In *Proceedings of the 16th Annual Conference of the European Association of Machine Translation, EAMT11*, Reykjavík, Iceland.

Forcada, Mikel L., Francis M. Tyers, and Gema Ramírez-Sánches. 2009. The Apertium machine translation platform: Five years on. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*, Alacant, Spain.

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

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

Loftsson, Hrafn and Eiríkur 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.

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

Pind, Jörgen, Friðrik Magnússon, and Stefán Briem. 1991. *Íslensk orðtíðnibók [The Icelandic Frequency Dictionary]*. The Institute of Lexicography, University of Iceland, Reykjavik.