# IceParser: An Incremental Finite-State Parser for Icelandic

**Hrafn Loftsson**
Department of Computer Science
Reykjavik University
Reykjavik, Iceland
hrafn@ru.is

**Eiríkur Rögnvaldsson**
Department of Icelandic
University of Iceland
Reykjavik, Iceland
eirikur@hi.is

## Abstract

We describe and evaluate an incremental finite-state parser for Icelandic – the first parser published for the language. Input to the parser is POS tagged text and it generates output according to a shallow syntactic annotation scheme, specifically designed for this project. The parser consists of a phrase structure module and a syntactic functions module. Both modules comprise a sequence of finite-state transducers, each of which adds syntactic information into substrings of the input text. F-measure for constituents and syntactic functions is 96.7% and 84.3%, respectively. These results are good, because Icelandic has a relatively free word order which can be difficult to account for in a parser. Moreover, of the various morphological features available in the rich POS tags, the transducers only use the case feature in their patterns.

## 1 Introduction

Syntactic analysis for natural languages is often divided into two categories: *full parsing*, in which a complete analysis for each sentence is computed, and *shallow parsing*, where sentence parts or chunks are analysed without building a complete parse tree.

One problem with full parsing is that the set of solutions can grow exponentially, because, generally, the parser considers all possible analysis of a given sentence. Moreover, since the goal is to build a complete parse tree for each sentence, the parser sometimes rejects a correct analysis of a sentence part on lower levels in the parse tree, on the ground that it does not fit into a global parse. Shallow parsing techniques do not have these problems because their aim is "to recover syntactic information efficiently and reliably from unrestricted text, by sacrificing completeness and depth of analysis" (Abney, 1996).

In many natural language processing (NLP) applications, it can be sufficient to analyse sentence parts or phrases. This can be the case, for example, in applications like information extraction, text summarisation and some types of grammar checking, in which identification of phrases is more important than a global parse. Additionally, in cases of low quality input or spoken language, a shallow parsing method can be more robust than a full parsing method, because of noise, missing words and mistakes in the input (Li and Roth, 2001).

In this paper we describe a shallow parser, *IceParser*, for parsing Icelandic text – the first parser published for the language[1]. *IceParser* is based on the incremental finite-state approach, in which a parser comprises a sequence of finite-state transducers. The transducers add syntactic information into the text in an incremental manner.

The input to *IceParser* is part-of-speech (POS) tagged text, using the detailed *IFD* tagset (Pind et al., 1991). It produces output according to a shallow annotation scheme, specifically designed for

[1]This work was partly supported by the Icelandic Research Fund, grant "Shallow parsing of Icelandic text".

**Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit (Eds.)**
**NODALIDA 2007 Conference Proceedings, pp. 128–135**

this project. The scheme consists of descriptions for annotation of both constituent structure and syntactic functions. Accordingly, the parser comprises two main modules: a phrase structure module and a syntactic functions module.

Evaluation shows that *IceParser* is both effective and efficient. F-measure for constituents and syntactic functions is 96.7% and 84.3%, respectively. These results are good, because the free word order in Icelandic can be difficult to account for in a parser. The parser is implemented in Java and processes about 11,300 word-tag pairs per second.

The remainder of this paper is organised as follows. In Section 2, we describe finite-state parsing in more detail. In Section 3, the main relevant features of Icelandic morphology and syntax are briefly described. Our annotation scheme is described in Section 4. We describe the design of *IceParser* in Section 5, and, in Section 6, we present the evaluation results. In Section 7, we analyse some of the errors, and we conclude in Section 8.

## 2 Finite-state parsing

Non-recursive language models, like finite-state grammars, have been used successfully to produce shallow parsers from the early 1990's.

The *reductionist* method by Koskenniemi et al. (1992) was influenced by the Constraint Grammar approach (Karlsson et al., 1995), in which syntactic tags are associated with words, instead of using phrase tree structures to represent parses. The main idea is to reduce all possible readings of a sentence (represented by finite-state automata) to one correct reading by a set of elimination rules.

A contrasting method is the *constructive* approach, which is based on a lexical description of a collection of syntactic patterns. The use of finite-state transducers to introduce syntactic labels into the input sentences is one example of the constructive approach.

A common constructive approach is to string together a sequence of transducers to build incremental (or cascading) shallow parsers (Grefenstette, 1996; Abney, 1997). Each transducer adds syntactic information into the text, such as brackets and names for grammatical functions. The hybrid method by (Aït-Mokhtar and Chanod, 1997) merges the constructive and the reductionist approaches by defining chunks (core phrases) by constraints rather than syntactic patterns.

The Xerox Finite-State Tool (XFST) (Karttunen et al., 1996) is often used to develop finite-state parsers. The XFST includes extensions to the standard regular expression calculus, which simplify the creation of finite-state transducers for syntactic processing.

Finite-state parsing methods have been used to develop a number of shallow parser for different languages, e.g. Spanish (Molina et al., 1999), Swedish (Megyesi and Rydin, 1999; Kokkinakis and Johansson-Kokkinakis, 1999), German (Müller, 2004), and French (Aït-Mokhtar and Chanod, 1997). Parsers built using finite-state methods are usually robust and fast, because they are, in fact, just a pipeline of lexical analysers.

## 3 Icelandic

Compared to its closest relatives (i.e. the other Nordic languages), Icelandic is a heavily inflected language, with nouns belonging to one of three genders, inflecting for four cases and two numbers. Additionally, nouns sometimes have a suffixed definite article. Adjectives inflect for four cases, three genders, three degrees, and two numbers, besides having both a "strong" (indefinite) and "weak" (definite) form; verbs inflect for three persons, two (main) moods, two tenses, and two voices; and so on.

Thus, the main Icelandic tagset, constructed in the compilation of the *IFD* corpus, is large (about 660 tags) compared to related languages. In this tagset, 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.

To illustrate, consider the word "*hestarnir*" (horses). The corresponding tag is "*nkfng*", denoting noun (*n*), masculine (*k*), plural (*f*), nominative (*n*), and suffixed definite article (*g*).

Due to the rich inflections which serve to indicate sentence-internal relationships and dependencies, Icelandic word order is rather free, especially

in many styles of written language. This freedom mainly concerns the relative order of major syntactic constituents, such as noun phrases (subjects and objects), preposition phrases, and adverb phrases, but also, to a certain extent, phrase-internal word order.

If we were aiming at full parsing and wanted to build a complete hierarchical parse tree, we would be faced with many difficult practical and theoretical questions. However, our annotation scheme is shallow in the sense that its syntactic structures are rather flat and simple, i.e. the main emphasis is to annotate core phrases without showing a complete parse tree. Therefore, the relatively free word order does not necessarily pose a problem for our syntactic annotation by itself, even though it may in many cases make it difficult for our parser to correctly identify certain syntactic constituents and especially syntactic functions.

## 4 The annotation scheme

Our annotation scheme follows the dominant paradigm in treebank annotation, i.e. it is "the kind of theory-neutral annotation of constituent structure with added functional tags" (Nivre, 2002).

Two labels are attached to each marked constituent. The first one denotes the beginning of the constituent, the second one denotes the end (e.g. *[NP ... NP]*). The main labels are *AdvP*, *AP*, *NP*, *PP* and *VP* – the standard labels used for phrase annotation (denoting adverb, adjective, noun, preposition, and verb phrase, respectively). Additionally, we use the labels *CP*, *SCP*, *InjP*, and *MWE* for marking coordinating conjunctions, subordinating conjunctions, interjections, and multiword expressions, respectively. Furthermore, we use the labels *APs* and *NPs*, for marking a sequence of adjective phrases (agreeing in gender, number and case) and noun phrases (agreeing in case), respectively.

Our scheme subclassifies VPs. A finite verb phrase is labelled as *[VP ... VP]* and consists of a finite verb, optionally followed by a sequence of AdvPs and supine verbs. Other types of VPs are labelled as *[VPx ... VPx]*, where x can have the following values: *i*, denoting an infinitive VP; *b*, denoting a VP which demands a predicate nominative (i.e primarily a verb phrase consisting of the verb *"vera"* (be)); *s*, denoting a supine VP; *p*, denoting a past participle VP; *g*: denoting a present participle VP.

We use curly brackets for denoting the beginning and the end of a syntactic function (as carried out by Megyesi and Rydin (1999)). Special function tags are used for labels: **QUAL*, **SUBJ*, **OBJ*, **OBJAP*, **OBJNOM*, **IOBJ*, **COMP*, **TIMEX*, denoting a genitive qualifier, a subject, an object, an object of an AP, a nominative object, an indirect object, a complement, and a temporal expression, respectively.

Additionally, for some of the syntactic function labels (see table 2), we use relative position indicators ("<" and ">"). For example, **SUBJ>* means that the verb is positioned to the right of the subject, **SUBJ<* denotes that the verb is positioned to the left, while **SUBJ* is used when it is not clear where the accompanying verb is positioned or when the verb is missing. The motivation behind using the indicators is to simplify grammar checking at later stages. A thorough description of the annotation scheme can be found in (Loftsson and Rögnvaldsson, 2006).

We have constructed a *grammar definition corpus* (GDC), a corpus consisting of 214 sentences (selected from the *IFD* corpus), representing the major syntactic constructions in Icelandic. The purpose of the GDC is to "provide an unambiguous answer to the question how to analyse any utterance in the object language" (Voutilainen, 1997). Furthermore, this corpus has been used as the development corpus for *IceParser*.

To illustrate the annotation scheme, consider the following sentence parts (shown without POS tags), obtained from the GDC.

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

- {*SUBJ> [NP systir NP] {*QUAL [NP hennar NP] *QUAL} *SUBJ>} [VPb var VPb]
  (sister her was)

- [VPb er VPb] {*SUBJ< [NP ég NP] *SUBJ<} {*COMP< [VPp fædd VPp] [CP og CP] [VPp uppalin VPp] *COMP<}
  (am I born and raised)

- {*SUBJ> [NP ég NP] *SUBJ>} [VPb er VPb] {*COMP< [AP bundin AP] *COMP<} {*OBJAP< [NP Reykjavík NP] *OBJAP<}
  (I am bound [to] Reykjavik)

## 5 IceParser

*IceParser* is designed to produce annotations according to the annotation scheme described in Section 4. The parser, which is purely constructive, consists of two main components: a phrase structure module (14 transducers) and a syntactic functions module (8 transducers). The purpose of the modular architecture "is to facilitate the work during development, to allow different uses of the parser and to reflect the different linguistic knowledge that is built into the parser" (Megyesi and Rydin, 1999). In both modules, the output of one transducer serves as the input to the following transducer in the sequence.

The transducers include numerous syntactic patterns, written to account for the relatively free word order of Icelandic. Apart from relying on *word class* or *subclass* information in the POS tags, the patterns only use the grammatical *case* feature.

The reason for not using to full extent the morphological information available in each POS tag, is that we want our parser to be utilised as a grammar checking tool, among other things. If the parser, for example, uses feature agreement to a great extent to mark phrases then it will not be possible for the grammar checking tool to point out feature agreement errors inside phrases. This is because the corresponding words would not have been recognised as one phrase by the parser, due to the lack of feature agreement!

The parser is implemented in Java and the lexical analyser generator tool JFlex (http://jflex.de/). Each transducer is written in a separate file, which is compiled into Java code using JFlex. The resulting Java code is a deterministic finite-state automaton (DFA), along with actions to execute for each recognised pattern. The actions add syntactic information into the text.

The reason for not using the XFST for implementation is that *IceParser* is part of a NLP toolkit for Icelandic, all of which is implemented in Java.

### 5.1 The phrase structure module

The purpose of the phrase structure module is to add brackets and labels to input sentences to indicate constituent structure. The syntactic annotation is performed in a bottom-up fashion, i.e. the deepest constituents are analysed first. For example, AdvPs are marked before APs, which are in turn marked before NPs.

To illustrate, consider the *Phrase_AdvP* transducer, which marks AdvPs, consisting of a single adverb, by putting the markers *[AdvP ... AdvP]* around it. An adverb in the input text is recognised using the regular expressions:

```
Adv={WordSpaces}{AdvTag}
```

The pattern *{WordSpaces}* denotes a sequence of word characters (all possible characters except white space) followed by one or more spaces. *{AdvTag}* is a pattern which matches an adverb POS tag. An *{Adv}* is thus a word tagged as an adverb.

The action associated with the *{Adv}* pattern is responsible for putting the appropriate brackets and labels around the recognised substring. For example, the word-tag pair *mjög aa* (very) is annotated as *[AdvP mjög aa AdvP]* by this transducer.

Consider the *Phrase_AP* transducer (slightly simplified), which marks APs, (using *[AP ... AP]*), consisting of a single adjective optionally preceded by an AdvP. It uses the following regular expressions:

```
Adj={WordSpaces}{AdjTag}
OpenAdvP="[AdvP"
CloseAdvP="AdvP]"
AdvPhrase={OpenAdvP}~{CloseAdvP}
AdjPhrase={AdvPhrase}?{Adj}
```

Here *{Adj}* is a pattern which matches a word tagged as an adjective. In JFlex, the regular expression $\sim a$ matches everything up to (and including) the first occurrence of a text matched by $a$. Thus, an *{AdvPhrase}*, to be included in an *{AdjPhrase}*, consist of a bracket and a label denoting the start of an adverb phrase followed by everything up to a label and a bracket denoting the end of the AdvP. For example, the substring *[AdvP mjög aa AdvP] góður lkensf* (very good) is annotated as *[AP [AdvP mjög aa AdvP] góður lkensf AP]* (henceforth, we do not show the POS tags in the examples).

The most complicated of all the transducers is the *Phrase_NP* transducer, which marks noun phrases

(the resulting DFA consists of about 50,000 states). This is due to the various ways a NP can be generated – from a single pronoun (e.g. *[NP hann NP]* (he)), to a sequence of an indefinite pronoun, a demonstrative pronoun/article, a numeral, an adjective phrase and a noun (e.g. *[NP allir þessir þrír [AP stóru AP] strákar NP]* (all these three big boys)).

For example, the substring *[AP [AdvP mjög AdvP] góður AP] kennari* (very good teacher) is annotated as *[NP [AP [AdvP mjög AdvP] góður AP] kennari NP]* by this transducer.

### 5.2 The syntactic functions module

The purpose of the syntactic functions module is to add tags to denote grammatical functions. The input to the first transducer in this module is the output of the last transducer in the phrase structure module.

To illustrate, consider the *Func_SUBJ* transducer, which annotates subjects. This transducer uses various patterns to recognise subjects, depending on whether the subject appears to the left of the finite verb phrase, to the right of the verb phrase, precedes a relative conjunction, etc. Here, we only discuss (a simplified version of) the main case, in which the subject appears to the left of the finite verb phrase:

```
NomSubj={NPNom}|{NPsNom}
VPorVPBe={VP}|{VPBe}
SubjVerb=({NomSubj}{WS}+{VPorVPBe}|
  {DatSubj}{WS}+{VPDat}|
  {AccSubj}{WS}+{VPAcc}
```

*{WS}+* denotes one or more white spaces. *{NomSubj}*, *{AccSubj}* and *{DatSubj}* match a single nominative NP, or a sequence of NPs, in the nominative, accusative, or dative case, respectively. *{VPorVPBe}* matches a finite verb phrase or a verb phrase containing the verb *"vera"* (be), and *{VPDat}* and *{VPAcc}* match verbs that demand oblique case subjects (this list of verbs is implemented as regular expressions). The action, associated with the pattern *{SubjVerb}*, finds out where the VP starts (using string searches) and puts the appropriate markers *{*SUBJ> ... *SUBJ>}* around the subject.

As another example, consider the *Func_COMP* transducer, whose main function is to annotate complements of the verb *"vera"*. A part of the patterns used by this transducer is (slightly simplified):

```
Compl={APSeqNom}|{NPSeqNom} |
      {VPPastSeq}
SubjVerbBe={Subject}{WS}+{VPBe}{WS}+
SubjVerbCompl={SubjVerbBe}{Compl}
```

According to this pattern a *{Compl}* is a sequence of nominative APs or NPs, or a sequence of past participle VPs. A *{SubjVerbBe}* is a *{Subject}*, followed by a verb phrase containing the verb *"vera"*. The action, associated with the pattern *{SubjVerbCompl}*, finds out where the VP ends and puts the appropriate markers *{*COMP< ... *COMP<}* around the complement.

For example, for the substring *[NP hann NP] [VPb er VPb] [NP [AP [AdvP mjög AdvP] góður AP] kennari NP]* (he is very good teacher), the application of the *Func_SUBJ* and *Func_COMP* transducers results in the string *{*SUBJ> [NP hann NP] *SUBJ>} [VPb er VPb] {*COMP< [NP [AP [AdvP mjög AdvP] góður AP] kennari NP]* *COMP<}.

## 6 Evaluation

*IceParser* has been evaluated on 509 sentences (8281 tokens), randomly selected from the *IFD* corpus. Since this corpus is only POS tagged, two annotators manually annotated the sentences (after the parser had been developed) with constituent structure and syntactic functions, according to our annotation scheme. The resulting treebank is our *gold standard*.

We used the *Evalb* bracket scoring program (Sekine and Collins, 1997) for automatic evaluation. For the evaluation of labelled constituent structure, we carried out two experiments. In the first one, we used the tags from the *IFD* corpus, i.e. we assumed correct tagging (see column 2 in table 1). In this case, the overall F-measure ($2 * precision * recall/(precision + recall)$) is 96.7%.

As can be deduced from table 1, *VP*, *CP*, *SCP* and *InjP* are "easy" to annotate. These phrase types constitute 28.6% of the phrases in the gold standard, and, thus, help to make the overall accuracy quite high. On the other hand, the accuracy for the more "difficult" phrase types, like *AP*, *NP* and *PP* (which constitute 58.7% of the phrases), is about 95%-97%, according to our results.

In the second experiment, we used the tagger *IceTagger* (Loftsson, 2006) to tag the sentences in the gold standard, before *IceParser* was run. The POS

| Phrase type | F-measure using correct POS tags | F-measure using *IceTagger* | Freq. in test data |
|---|---|---|---|
| AdvP | 91.8% | 85.1% | 8.2% |
| AP | 95.1% | 86.3% | 8.1% |
| APs | 87.0% | 68.6% | 0.5% |
| NP | 96.8% | 93.0% | 37.6% |
| NPs | 80.4% | 74.3% | 1.5% |
| PP | 96.7% | 91.3% | 13.0% |
| VPx | 99.2% | 93.8% | 19.3% |
| CP | 100.0% | 99.6% | 5.7% |
| SCP | 99.6% | 97.6% | 3.4% |
| InjP | 100.0% | 96.3% | 0.2% |
| MWE | 96.9% | 92.6% | 2.5% |
| All | 96.7% | 91.9% | 100.0% |

Table 1: Results for the various phrase types.

| Function type | F-measure using correct POS tags | F-measure using *IceTagger* | Freq. in test data |
|---|---|---|---|
| SUBJ | 68.2% | 47.6% | 4.7% |
| SUBJ> | 92.7% | 89.4% | 30.3% |
| SUBJ< | 83.7% | 75.1% | 12.3% |
| OBJ | 0.0% | 0.0% | 0.2% |
| OBJ> | 43.5% | 20.0% | 0.8% |
| OBJ< | 90.2% | 78.2% | 19.7% |
| OBJAP> | 71.4% | 57.2% | 0.2% |
| OBJAP< | 75.0% | 46.2% | 0.4% |
| OBJNOM< | 30.8% | 16.7% | 0.6% |
| IOBJ< | 73.3% | 51.9% | 0.9% |
| COMP | 56.9% | 40.0% | 2.8% |
| COMP> | 91.3% | 91.3% | 1.3% |
| COMP< | 75.1% | 70.0% | 12.7% |
| QUAL | 87.7% | 77.9% | 10.4% |
| TIMEX | 74.7% | 55.9% | 2.7% |
| All | 84.3% | 75.3% | 100.0% |

Table 2: Results for the various syntactic functions.

tagging accuracy for these sentences is 91.1% (unknown word ratio is 7.8%). In this case, the overall F-measure for constituent structure drops from 96.7% to 91.9% (see column 3 in table 1), which is equivalent to about 5.0% reduction in accuracy. The POS tagging accuracy is relatively low, compared to related languages, and this has substantial effect on the overall parsing accuracy.

Unfortunately, we can not compare our results to other parsers for Icelandic, since this evaluation is the first parser evaluation published for the language. For the sake of a comparison with a related language[2], Swedish, Knutsson et al. (2003) report 88.7% F-measure for all phrases, and 91.4% for NPs, when using a tagger to preprocess the text and a shallow (not finite-state) rule-based parser. Using a finite-state parser, Kokkinakis and Johansson-Kokkinakis (1999) report higher numbers, 93.3% for all phrases and 96.2% for NPs, despite using a tagger for preprocessing. The tagger used, however, obtains very high accuracy when tagging the test data, i.e. 98.7%. We believe that this comparison indicates our parser performs well when annotating constituents.

For the evaluation of syntactic functions, we also carried out experiments with and without correct tagging. When using the correct tags from the *IFD* corpus, the overall F-measure is 84.3% – see column 2 of table 2. When considering subjects and objects, the highest accuracy is obtained for the functions *SUBJ>* and *OBJ<*, i.e. a subject whose accompanying verb is to the right, and an object whose accompanying verb is to the left. This was to be expected, because the normal word order is SVO. If the relative position indicator is ignored (many shallow parser do not include such an indicator), thus, for example, combining the three subject functions into one, F-measure for *SUBJ* and *OBJ* is 90.5% and 88.2%, respectively.

When *IceTagger* is used to produce tags, the overall F-measure for syntactic functions drops from 84.3% to 75.3% (see column 3 in table 2), which is equivalent to about 10.7% reduction in accuracy. Thus, the accuracy of the syntactic functions module is more sensitive to tagging errors than the constituent module. This can be explained by the fact that the former component relies to a much higher extent on the case feature, which is often responsible for the errors made by the tagger.

Again, we are not in a position to compare our results to another Icelandic parser. For German (a

[2]Note that comparison between languages is questionable, because of different language characteristics, parsing methods, annotation schemes, test data, evaluation methods, etc.

related language), Müller (2004), for example, has presented the following results of syntactic function annotation using a finite-state parser (and POS tags from a corpus): 82.5% F-measure for all functions, and 90.8%, 64.5% and 81.9%, for subjects, accusative objects and dative objects, respectively. If these results are used for comparison, *IceParser* seems to obtain good results for syntactic functions.

In the first version of *IceParser*, the output file of one transducer is used as an input file in the next transducer in the sequence. This version processes about 6,700 word-tag pairs per second (running on a Dell Optiplex GX620 Pentium 4, 3.20 GHz). We have implemented another version of the parser which, instead of reading and writing to files, reads from and writes directly to memory (using the Java classes *StringReader* and *StringWriter*). This version annotates about 11,300 word-tag pairs per second, which is equivalent to about 75% speed increase compared to the previous version.

## 7 Error analysis

In this section, we show examples of the errors made by *IceParser*.

The only type of error in adverb phrase annotation occurs when the parser incorrectly groups together two (or more) adjacent adverbs. Consider the incorrect output:
*[PP um [NP það NP] PP] [VP vissi VP] [NP stelpan NP] [AdvP ekki þá AdvP]* (about that knew girl not then).
The two adverbs at the end should form two distinct AdvPs, because *"ekki"* is a sentence adverb which does not modify the temporal adverb *"þá"*.

Adverbs are also the source of some of the errors made in then annotation of adjective phrases. Consider the incorrect output:
*[CP og CP] [VP tóku VP] [NP [AP [AdvP fram AdvP] eigin AP] dósir NP]* (and took out own cans). In this sentence part, the adverb *"fram"* is a particle associated with the verb *"tóku"* (take out), but not a modifier of the adjective *"eigin"*.

A frequent noun phrase error made by the parser is exemplified by the incorrect output *[NP árin NP] [AP gullnu AP]* (years golden). Here, the correct annotation is *[NP árin [AP gullnu AP] NP]*, because the adjective *"gullnu"* is a post-modifier of the noun *"árin"*. *IceParser* makes this type of error, because it does not include a pattern for noun-adjective word order.

The *Phrase_NPs* transducer groups together a sequence of noun phrases agreeing in case. Consider the incorrect output:
*[AP sterkur AP] [VPb var VPb] [NPs [NP hann NP] [CP og CP] [NP íþróttamaður NP] NPs] [AP ágætur AP]* (strong was he and athlete fine).
Here, the parser groups the noun phrases *[NP hann NP] and [NP íþróttamaður NP]* together, because the phrases agree in case. The correct annotation, however, is:
*[AP sterkur AP] [VPb var VPb] [NP hann NP] [CP og CP] [NP íþróttamaður [AP ágætur AP] NP].*

To give an example where *IceParser* annotates a subject without a correct position indicator, consider the output:
*[VPb er VPb] [AdvP ekki AdvP] [VPi að koma VPi] {*SUBJ [NP matur NP] *SUBJ}?* (is not to come food?).
The correct annotation for the subject is *{*SUBJ< [NP matur NP] *SUBJ<}*, because *"matur"* is the subject of the verb *"er"* at the beginning of the sentence. *IceParser* does include patterns to match the order VP-AdvP-SUBJ, but, in this case, the infinitive verb phrase *[VPi að koma VPi]* is positioned in between the AdvP and the SUBJ. The parser, however, marks all stand-alone nominative NPs as *{*SUBJ ... *SUBJ}*, and therefore the *[NP matur NP]* phrase does receive subject marking, albeit incomplete.

Finally, note that some of the errors in syntactic function annotation are due to errors made in the phrase structure annotation. For the incorrect phrase structure output (discussed above) *[CP og CP] [VP tóku VP] [NP [AP [AdvP fram AdvP] eigin AP] dósir NP]*, the parser will produce the syntactic function:
*{*OBJ< [NP [AP [AdvP fram AdvP] eigin AP] dósir NP] *OBJ<}*.
This object is incorrect, because it includes the adverb phrase *[AdvP fram AdvP]*.

## 8 Conclusion

We have described and evaluated the incremental finite-state parser *IceParser*, for parsing Icelandic

text. The parser comprises two modules: a phrase structure module and a syntactic functions module. Both modules consist of a sequence of transducers, which add syntactic information into the input strings, according to our shallow syntactic annotation scheme.

Evaluation shows that F-measure for phrases and syntactic functions is 96.7% and 84.3%, respectively. We have argued that these results are good, because Icelandic has a relatively free word order, which is difficult to account for in a parser. Moreover, of the various morphological features available in the rich POS tags, the transducers only use the case feature in their patterns.

In future work, we would like to improve individual components of our parser, and build a version of it which utilises to a greater extent the morphological information available in the POS tags.

## 9 Acknowledgements

Thanks to the Institute of Lexicography at the University of Iceland, for providing access to the *IFD* corpus.

## References

S. Abney. 1996. Part-of-Speech Tagging and Partial Parsing. In K. Church, S. Young, and G. Bloothooft, editors, *Corpus-Based Methods in Language and Speech*. Kluwer Academic Publishers.

S. Abney. 1997. Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2(4):337–344.

S. Aït-Mokhtar and J-P. Chanod. 1997. Incremental Finite-State Parsing. In *Proceedings of Applied Natural Language Processing*, Washington DC, USA.

G. Grefenstette. 1996. Light Parsing as Finite State Filtering. In *Proceedings of the ECAI '96 workshop on "Extended finite state models of language"*, Budapest, Hungary.

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

L. Karttunen, J-P. Chanod, Grefenstette, G., and A. Schiller. 1996. Regular expressions for language engineering. *Natural Language Engineering*, 2(4):305–328.

O. Knutsson, J. Bigert, and V. Kann. 2003. A Robust Shallow Parser for Swedish. In *Proceedings of NoDaLiDa 2003*, Reykjavik, Iceland.

D. Kokkinakis and S. Johansson-Kokkinakis. 1999. A Cascaded Finite-State Parser for Syntactic Analysis of Swedish. In *Proceedings of the* $9^{th}$ *Conference of the European Chapter of the ACL (EACL)*, Bergen, Norwsay.

K. Koskenniemi, P. Tapanainen, and A. Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the* $14^{th}$ *International Conference on Computational Linguistics*, Nantes, France.

X. Li and D. Roth. 2001. Exploring Evidence for Shallow Parsing. In *Proceedings of the* $5^{th}$ *Conference on Computational Natural Language Learning*, Toulouse, France.

H. Loftsson and E. Rögnvaldsson. 2006. A shallow syntactic annotation scheme for Icelandic text. Technical Report RUTR-SSE06004, Department of Computer Science, Reykjavik University.

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

B. Megyesi and S. Rydin. 1999. Towards a Finite-State Parser for Swedish. In *Proceedings of NoDaLiDa 1999*, Throndheim, Norway.

A. Molina, F. Pla, L. Moreno, and N. Prieto. 1999. APOLN: A Partial Parser of Unrestricted Text. In *Proceedings of SNRFAI99*, Bilbao, Spain.

F-H. Müller. 2004. Annotating Grammatical Functions in German Using Finite-State Cascades. In $20^{th}$ *International Conference on Computational Linguistics*, Geneva, Switzerland.

J. Nivre. 2002. What kinds of trees grow in Swedish soil? A Comparison of Four Annotation Schemes for Swedish. In *Proceedings of the* $1^{st}$ *Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

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

S. Sekine and M.J. Collins. 1997. The Evalb Software. http://nlp.cs.nyu.edu/evalb/. Site visited November 2006.

A. Voutilainen. 1997. Designing a (Finite-State) Parsing Grammar. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*. MIT Press.