Adapting and Improving a Shallow Parser

by

Gunnar Pálsson

Thesis of 30 ECTS credits submitted to the School of Technology,
Department of Computer Science
at Reykjavík University in partial fulfillment
of the requirements for the degree of
Master of Science (M.Sc.) in Language Technology

June 2021

Examining Committee:

Hrafn Loftsson, Supervisor
Associate Professor, Reykjavík University, Iceland

Hannes Högni Vilhjálmsson, Examiner
Professor, Reykjavík University, Iceland

Yngvi Björnsson, Examiner
Professor, Reykjavík University, Iceland

Adapting and Improving a Shallow Parser

Gunnar Pálsson

June 2021

Abstract

IceNLP is a set of tools for processing and analysing the Icelandic language. One of these tools is IceParser, a shallow parser, whose purpose is to annotate flat phrase structure and syntactic functions. The input to IceParser are tokenised sentences, tagged with part-of-speech (POS) tags from the Icelandic POS tagset. This tagset was revised recently which means that many tools need to be revised to accurately handle the changes. In this project, we, first, present the changes made to IceParser with regard to the tagset revision. Second, we describe improvements made to individual components of IceParser, for the purpose of increasing its precision and coverage. Third, we perform an evaluation of the functionality of IceParser after the changes and, finally, perform preliminary error analysis.

Keywords: Icelandic, Part-of-Speech Tagging, Shallow Parsing, IceNLP

Aðlögun og umbætur á hlutaþáttara

Gunnar Pálsson

júní 2021

Útdráttur

IceNLP er safn hugbúnaðartóla til að vinna með og greina íslenskan texta. Eitt þessara tóla er IceParser, hlutaþáttari með þann tilgang að greina setningarliði og setningafræðileg hlutverk. Inntak IceParser eru tilreiddar setningar sem hafa verið markaðar samkvæmt íslenska markamenginu. Íslenska markamengið var nýlega uppfært og þurfa tól IceNLP uppfærslu til að taka á breytingum markamengisins vegna þess. Í þessari ritgerð kynnum við breytingar sem hafa verið gerðar á IceParser. Við kynnum einnig almennar umbætur á einstökum hlutum IceParser sem eru gerðar í þeim tilgangi að bæta nákvæmni og griphlutfall. Við framkvæmum mælingu á virkni IceParser eftir breytingarnar og að lokum framkvæmum við villugreiningu.

Efnisorð: Íslenska, Mörkun, Hlutaþáttun, IceNLP

Adapting and Improving a Shallow Parser

Gunnar Pálsson

Thesis of 30 ECTS credits submitted to the School of Technology,
Department of Computer Science
at Reykjavík University in partial fulfillment of
the requirements for the degree of
Master of Science (M.Sc.) in Language Technology

June 2021

Student:

.....
Gunnar Pálsson

Examining Committee:

.....
Hrafn Loftsson

.....
Hannes Högni Vilhjálmsson

.....
Yngvi Björnsson

.....
date

.....
Gunnar Pálsson
Master of Science

I dedicate this to friends, family, and all those
that helped me in these turbulent times

Acknowledgements

This project was funded by the Language Technology Programme for Icelandic 2019-2023. The programme, which is managed and coordinated by Almannarómur[1], is funded by the Icelandic Ministry of Education, Science and Culture.

Additional thanks to Hulda Óladóttir of Miðeind ehf., for providing us with the linguistic corpus which was the basis for our evaluations, and Haukur Páll Jónsson, for tagging the corpus with a state-of-the-art tagger currently being worked on.

[1] https://almannaromur.is/

Preface

This dissertation is original work by the author, Gunnar Pálsson, with assistance and supervision from Dr. Hrafn Loftsson, who also helped with the manual annotation of the gold standard.

Contents

List of Figures

List of Tables

List of Abbreviations

CLARIN-IS	Common Language Resources & Technology Infrastructure - Iceland
DFA	Deterministic Finite Automata
IFD	Icelandic Frequency Dictionary
IGC	Icelandic Gigaword Corpus
LT	Language Technology
MIM	Mörkuð íslensk málheild (e. Tagged Icelandic Corpus)
NLP	Natural Language Processing
POS	Part of Speech
SVO	Subject-Verb-Object
F_1	Traditional F-measure
Pr	Precision
Re	Recall

AP	Adjective Phrase
APs	Adjective Phrases
AdvP	Adverb Phrase
CP	Conjunction Phrase
FRW	Foreign Word
FRWs	Foreign Words
InjP	Interjection Phrase
MWE	Multi-Word Expression
NP	Noun Phrase
NPs	Noun Phrases
PP	Preposition Phrase
SCP	Subordinating Conjunction Phrase
VP	Verb Phrase
VPb	Verb Phrase (*be*-verb)
VPi	Verb Phrase (infinitive)
VPg	Verb Phrase (present participle)
VPp	Verb Phrase (past participle)
VPx	Verb Phrase (all)
COMP	Complement
IOBJ	Indirect Object
OBJ	Object
OBJAP	Object of Adjective
OBJNOM	Nominative Object
QUAL	Qualifier
SUBJ	Subject
TIMEX	Temporal Expression

Chapter 1

Introduction

The processing and analysis of text for the Icelandic language is a relatively new field. In the last 20 or so years, the field has bloomed and Icelandic now has many linguistic tools in its arsenal to stave off the so-called digital death [1]. Icelandic has been at risk of suffering a digital death due to its low number of native speakers, lack of resources, and low usage online [2]. Some of these tools have been compiled into a Natural Language Processing (NLP) toolkit known as IceNLP [3]. One of the tools in IceNLP is the first parser made for Icelandic, IceParser [4], which was released in 2007. IceParser's purpose is to take text and identify any and all phrases and syntactic functions within it.

IceParser depends on text being POS-tagged beforehand which, in most cases, means that any plain text must first be run through a POS tagger, of which IceNLP contains several. Within IceNLP, all tools employ POS tags of the same type, i.e. tags that conform to the detailed Icelandic tagset, a tagset created alongside the formation of the Icelandic Frequency Dictionary (IFD) [5]. The tagger gives each word its own tag. These tags contain important information about each word, such as the word's class and its morphological attributes. Each tag is a series of letters where each letter represents an attribute. An example of a standard tag and its makeup can be seen in Section 2.1. In short, a tagger tags plain text according to the Icelandic tagset, creating a tagged version of said text. That tagged text then gets parsed by IceParser, where all its phrases and functions are identified. The Icelandic tagset has gone through a recent revision which altered the way in which many word types are now tagged. This means that any tools that rely on it, e.g. taggers and IceParser, must receive modifications if they are to be useful.

By systematically implementing modifications which take tagset alterations into account, IceParser has been modified such that it is fully able to parse text that has been tagged with the newest version of the Icelandic tagset. IceParser also retains the functionality to parse text tagged with older versions of the tagset, i.e. care was taken to avoid automatic deprecation of tools which use older tagset versions. Our evaluations show that IceParser's accuracy is comparable to (albeit slightly lower than) what the original evaluations indicate [4]. When given a 509 sentence test corpus, the newest version of IceParser shows an F_1 score of 96.34% for phrases and 83.09% for functions. Directly comparing it to the original IceParser evaluations from 2007, this is a performance drop of 0.35 and 1.21 percentage points, respectively.

This report is structured as follows: In Chapter 2, we go into the background and history of some of the primary components involved in the project. In Chapter 3, we cover all the modifications made to IceParser, grouping them into one of three categories based on their relation to the revised tagset. In Chapter 4, we briefly cover the methodology used to identify error patterns before presenting the evaluation results for IceParser. We also perform a comparison of the newest version of IceParser to the original IceParser and its accuracy. In Chapter 5, we discuss IceParser's evaluation results and other notable aspects of its parsing accuracy. In Chapter 6, we recall some of the major parsing errors known to persist, categorizing each by whether or not it existed before the modifications. Finally, in Chapter 7, we consider the implications of the test results and conclude.

Chapter 2

Background

2.1 IceNLP

IceNLP is an NLP toolkit made for Icelandic which contains many useful linguistic tools developed independently, including Lemmald [6], IceTagger [7] and IceParser [4], [8]. IceNLP and other linguistic resources are part of the CLARIN-IS LT repository for Icelandic [9]. There is a specified pipeline with which plain text is passed through where it is processed by each tool, but each tool can be used by itself as long as its input is consistent with the output of the previous tool in the pipeline, e.g. the tagger directly precedes IceParser in the pipeline so any input to IceParser must therefore match the output format of the tagger.

Two particular tools shall be highlighted: The tagger, one of several included in IceNLP, and IceParser, the focus of this project. The tagger's purpose is to take plain text and convert it to tagged text where each word is assigned a tag which represents the morphological attributes of said word. IceParser's purpose is to take tagged text and identify the phrases (e.g. noun phrases, verb phrases) and syntactic functions (e.g. subjects, objects) within the text.

The POS tags work like this: The first letter of a tag denotes the word class (e.g. **n** for nouns, **s** for verbs) and the rest of the letters denote the morphological attributes specific to each word form. For example, the tag for the word *hesturinn* (e. *the horse*) is **nkeng** as it is a noun (**n**) which is masculine (**k**), singular (**e**), in the nominative case (**n**), and has a suffixed definite article (**g**). The tagger places the tag between the tagged word and the next, essentially replacing each word with a (word, tag) pair.

A recent revision of the Icelandic tagset [10] meant that IceParser needed its own revision to accurately process the new tagset while also retaining functionality to process older versions of the tagset. This prevents automatic deprecation of any tools which use older versions of the tagset.

An example of how tagging works can be seen in Figure 2.1 where the sentence *Þetta er mjög gott* (e. *This is very good*) is run through an IceNLP tagger. The input is plain text and the output is tagged text where each word's tag is placed directly after the word it represents. The word tags are shown in red.

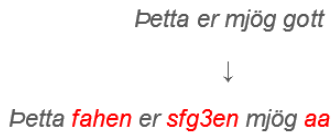

Figure 2.1: An example of an IceNLP tagger's input and output

2.2 IceParser

IceParser is a rule-based shallow parser [11] with a two-fold purpose: Identifying and annotating phrases and syntactic functions. IceParser is the first parser published for the Icelandic language, being released in 2007. IceParser being rule-based means that it uses pre-defined patterns to parse sentences and being a shallow parser means there is no hierarchy of functions that culminate in a function representing the whole sentence, i.e. no attempt is made in building a full parse tree.

IceParser works by identifying patterns of word sequences in sentences and assigning appropriate phrases or functions to those patterns. Icelandic is a morphologically rich language with very free word order. This makes functional word patterns highly variable and IceParser must therefore be highly varied in its pattern matching capabilities. With that said, Icelandic does have many fixed rules that can be exploited. Icelandic is, for example, a subject-verb-object (SVO) language where the subject is most commonly in the nominative case and the object is most commonly in an oblique case.

The input to IceParser is text which has been tagged according to the Icelandic tagset, a tagset originally based on the detailed IFD tagset. Originally, the tagset used was simply the IFD tagset itself, but later revisions resulted in the creation of a unique tagset, henceforth referred as the Icelandic tagset or simply the tagset. IceNLP contains several taggers, e.g. IceTagger [7] and IceStagger [12] all of which convert text to (word, tag) pairs. The output of these taggers is the most common source of input to IceParser.

The output of IceParser is quite easy to understand: Phrases are signified by square brackets and functions are signified by curly brackets. Each bracket starts with a tag identifying the type of phrase or function. For example, the parsed sentence *{*SUBJ> [NP þetta fahen] } [VPb er sfg3en] {*COMP< [AP gott lhensf] }* (e. *this is good*) contains a noun phrase (NP), a verb phrase (VPb; more specifically, a *be*-verb phrase), and an adjective phrase (AP). The noun phrase is identified as a subject and the adjective phrase as a complement. Note the arrows in the function tags, these point to the verb of the relevant statement. These arrows are optional and can be omitted when the main verb is not found, unclear, or missing. This structure, the functions in particular, is based on [13]. A thorough description of the annotation scheme, along with a complete description of all phrase and function tags, can be found in [14]. A complete list of phrase and function types IceParser can identify can be found in Section 2.2.1.

The data IceParser uses to match patterns are the tag part of the (word, tag) pairs in most cases. Before the alterations talked about in this paper, IceParser almost exclusively looked at the tags[1]. However, due to different behaviour of auxiliary verbs depending on whether they follow *be*-verbs or not, i.e. whether the auxiliary verb is a past participle or supine, this must be changed. As they are tagged identically, the words themselves must be checked to distinguish between *be*-verbs and other verbs. More on that in Sections 2.3.1 and 3.3.

An example of how IceParser's parsing works is shown in Figure 2.2 where the output from the tagging example shown in Figure 2.1 is run through IceParser. First, the phrases (whose tags are displayed in green) are bounded by square brackets and, second, the functions (whose tags are displayed in blue) are bounded by curly brackets.

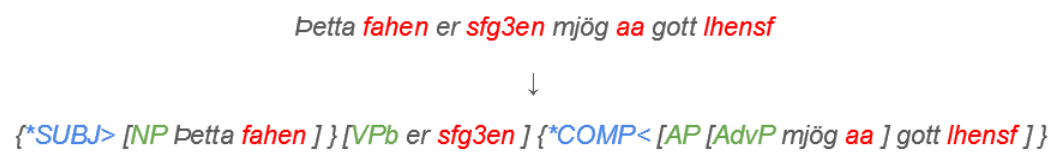

Figure 2.2: An example of IceParser's input and output

2.2.1 Phrase and Function Types

Tables 2.1 and 2.2 list the different types of phrases and functions which IceParser identifies with some examples in Icelandic. Most of the examples were found in [14] and in the gold standard defined in Section 4.1 and their tags have been omitted for simplicity. Phrases and non-example functions have also been omitted from the function examples for the same reason. Note that three phrase types (APs, NPs, and PP) are necessarily nested, i.e. they always contain at least one other phrase.

2.3 JFlex

Most of IceParser's functionality is integrated with JFlex [15], a lexical analyzer generator for Java. JFlex reads input in the form of regular expressions and corresponding actions. The result of compiling said input with JFlex is Java code in the form of a DFA [16], a common method of accepting or rejecting sequences of strings. The input is kept in .flex files, referred to as transducers in NLP, which contain a multitude of patterns which ultimately result in identification of phrases and functions.

IceParser's pattern matching functionality is split into multiple transducers where each one focuses on a single linguistic function. For example, one transducer only focuses on identifying noun phrases while another only focuses on identifying adjective phrases and yet another focuses on identifying subjects. IceParser's inner pipeline first identifies phrases by matching word and tag patterns and then identifies syntactic functions by matching phrase patterns. Most modifications to IceParser are made within these transducers but several changes involve moving regular expressions between transducers or into new ones. More on that in Section 3.3.

[1]Only one instance of the words themselves being identified instead of tags was in the previous version of IceParser. In that instance, the names of months were used to identify temporal expressions.

Tag	Phrase type	Example
AdvP	Adverb Phrase	[AdvP jafnvel]
AP	Adjective Phrase	[AP gott]
APs	AP sequence	[APs [AP margra] [AP rangra]]
CP	Conjunction Phrase	[CP og]
SCP	Subordinating CP	[SCP sem]
FRW	Foreign Word	[FRW balsamic]
FRWs	FRW sequence	[FRWs hate will prevail]
InjP	Interjection Phrase	[InjP Já]
NP	Noun Phrase	[NP formaður]
NPs	NP sequence	[NPs [NP Boeing] [CP og] [NP Airbus]]
PP	Preposition Phrase	[PP hjá [NP mér]]
VP	Verb Phrase	[VP kom]
VPb	*Be*-VP	[VPb er]
VPi	Infinitive VP	[VPi að ganga]
VPg	Present part. VP	[VPg varðandi]
VPp	Past part. VP	[VPp gert]
MWE_AdvP	Multi-Word AdvP	[MWE_AdvP hins vegar]
MWE_AP	Multi-Word AP	[MWE_AP annars konar]
MWE_CP	Multi-Word CP	[MWE_CP eins og]
MWE_PP	Multi-Word PP	[MWE_PP þrátt fyrir]

Table 2.1: All phrase types and examples

Tag	Function type	Example
COMP	Complement	Þetta er {*COMP< geðveikt }
IOBJ	Indirect Object	Gerðum {*IOBJ< okkur } grein fyrir því
OBJ	Object	Ég vona {*OBJ< það }
OBJAP	Object of Adjective	Safnið er opið {*OBJAP< fólki }
OBJNOM	Nominative Object	Þeim fannst {*OBJNOM< myndin } góð
QUAL	Qualifier	Þetta er sonur {*QUAL hans }
SUBJ	Subject	{*SUBJ> Við } ætluðum að vinna
TIMEX	Temporal Expression	Hún fæddist {*TIMEX árið 1982 }

Table 2.2: All function types and examples

2.3.1 Transducers

A transducer has two critical parts: Regular expressions, and corresponding actions. In Figure 2.3, a simplified path through a transducer[2] can be seen as a series of five steps. Specifically, steps one to four are regular expressions, and the fifth step is corresponding action. These steps are the general formula that most transducers have, with minor differences on occasion.

```
ArticleTag = g{Gender}{Number}{Case}
PossessivePronounTag = fe{Gender}{Number}{Case}
ReflexivePronounTag = fb{Gender}{Number}{Case}
...
Article = {WordSpaces}{ArticleTag}
PossessivePronoun = {WordSpaces}{PossessivePronounTag}
ReflexivePronoun = {WordSpaces}{ReflexivePronounTag}
...
ArticleNP = {Article}{Numeral}?{AdjectivePhrases}?{Noun}?
PossessiveNP = {PossessivePronoun}({AdjectivePhrases}?{Noun})?
ReflexiveNP = {ReflexivePronoun}({Noun}|{PersonalPronoun})?
...
NounPhrase = {ArticleNP} | {PossessiveNP} | {ReflexiveNP}
...
{MWE}           { out.write(str);}
{NounPhrase}    { out.write(NPOpen + str + NPClose);}
```

Figure 2.3: A snippet of code from Phrase_NP.flex

As in the example given in Figure 2.3, the first step is to identify a word's tag. During preprocessing, IceParser always starts by encoding the tags so as to not confuse words and tags. The terms inside curly brackets denote previously designated and appropriately named regular expression patterns[3]. The second step is to identify a word and its tag as a single entity. The third step is to identify the whole phrase or function. This step often introduces potential variables designated with question marks[4]. The fourth step is to group all patterns that should be treated equally by IceParser. This step is optional but preferable as it reduces the amount of variables in the fifth step. The fifth and final step is to assign actions to each pattern. In IceParser, this means placing brackets around the phrases and functions which have been identified and tagging said brackets appropriately.

Many transducers have built-in skips in the fifth step where the assigned action is to do nothing. This can be seen in Figure 2.3 where multi-word expressions (MWE) remain unchanged whether a pattern was identified or not. These skips are the primary reason for the order within IceParser's pipeline as any identified pattern can be easily ignored by any transducers down the line. This is also the method chosen to distinguish between *be*-verbs and other verbs. More on that in Section 3.3.

[1]Or any VP whose main verb demands a predicate nominative, e.g. *verða* (e. *become*) or *heita* (e. *name*).

[2]Specifically, Phrase_NP.flex, the primary transducer for identifying noun phrases.

[3]For example, {Case} is a letter which represents any of the four cases in Icelandic: **n** for nominative, **o** for accusative, **þ** for dative, or **e** for genitive.

[4]For example, noun phrases can optionally have adjectives in many cases.

2.3.2 The Transducer Pipeline

Let's take a simple sentence, *Ég tala góða ensku* (e. *I speak good English*), and make it go step-by-step through IceParser's transducer pipeline. Table 2.3 shows this journey in sequence, resulting in the fully parsed version of the sentence, *{*SUBJ> [NP Ég] } [VP tala] {*OBJ< [NP [AP góða] ensku] }*. Note that, under normal circumstances, the sentence would be tagged, but the tags have been omitted for the sake of simplicity. The tags would simply appear after the word it describes, within the same brackets.

Transducer	Result of transducer
Before	Ég tala góða ensku
Phrase_AP	Ég tala [AP góða] ensku
Phrase_NP	[NP Ég] tala [AP góða] ensku
Phrase_NP	[NP Ég] tala [NP [AP góða] ensku]
Phrase_VP	[NP Ég] [VP tala] [NP [AP góða] ensku]
Func_SUBJ	{*SUBJ> [NP Ég] } [VP tala] [NP [AP góða] ensku]
Func_OBJ	{*SUBJ> [NP Ég] } [VP tala] {*OBJ< [NP [AP góða] ensku] }

Table 2.3: The step-by-step journey of a sentence through IceParser

Chapter 3

Modifications

Modifications made to IceParser can be split into two groups: Necessary modifications due to tagset changes and modifications to expand the functionality of IceParser regardless of the new tagset. The first group can be further split into two more groups: Modifications needed because of changes to existing tags and modifications needed because of brand new tags. Also note that not all tagset changes are mentioned here, only the ones which required modifications to be made to IceParser.

The tagset-independent modifications can be viewed as actions made to increase performance and the tagset-dependent modifications as actions made to minimize performance drop due to the tagset revision.

Some of the tagset-independent modifications also have a secondary purpose: refactoring, or changing the internal structure of code without altering its external behaviour [17]. This is done to simplify the structure of IceParser's code, and to make locations of certain functionality of IceParser more intuitive.

Changes are made by changing the code within relevant transducers. One of the more simplistic modifications can be seen in Figures 3.1 and 3.2. These code snippets show the modifications to preposition tags mentioned in Section 3.1, specifically the additional identification of the preposition tag **af** alongside the exiosting preposition tags **ao**, **aþ**, and **ae**. In the example, the letters **o**, **þ**, and **e** are already identified as the pattern **{ObliqueCase}**. Note that no removal takes place, only addition. This is because previous functionality is retained.

```
PrepTag = {encodeOpen}a{ObliqueCase}[em]?{encodeClose}{WhiteSpace}+
```

Figure 3.1: A code snippet identifying preposition tags before modifications

```
PrepTag = {encodeOpen}a({ObliqueCase}|f)[em]?{encodeClose}{WhiteSpace}+
```

Figure 3.2: A code snippet identifying preposition tags after modifications

Most other modifications are more complex than this example, but they mostly follow the same method wherein existing lines of pattern-identifying code are modified to accept new patterns.

3.1 Changes to the Tagset

i) **Prepositions**

- **ao**, **aþ** and **ae** are now tagged as **af**

Prepositions were formerly tagged according to what case the preposition demanded (**o** for accusative, **þ** for dative, **e** for genitive). Specifying a preposition's case is considered redundant, the case is already specified by the preposition's object. A side effect of this redundancy is that errors got doubled as both the preposition and its object got marked as wrong. For example, the preposition phrase *um leikinn* (e. *about the game*) is in the accusative case. But, if it is incorrectly tagged as belonging to another case, both words are marked as wrong and two errors are identified. This was considered unnecessary and only one error should result from situations like this. To fix this, it was decided to simplify preposition tags. Now, all prepositions are simply tagged as **af**.

ii) **Verbs**

- **ssg** are now tagged as **sþghen**
- **ssm** are now tagged as **sþmhen**

Supine verbs formerly had a unique tag: **ss** followed by either **g** or **m** for the active or middle voice, respectively. Now, supine verbs are represented by a tag that fits the generalized scheme for verbs. This new tag for supine verbs is shared with past participle verbs which causes ambiguity when parsing. This means that IceParser must use other tactics to disambiguate the two because past participles can be complements, but supine verbs cannot. More on this in Section 6.2.

iii) **Nouns**

- Tags starting with **nx** now start with **n-**

The second letter of noun tags (which denotes the noun's gender) formerly had the option of a special tag (**x**) where the gender was defined as *Unspecified*. Now, this is replaced by a unique non-tag character (**-**) which was already in use in other situations.

iv) **Foreign Words**

- Foreign proper nouns are now tagged as **n----s**

Taggers formerly used to attempt to identify the morphological attributes of foreign names, causing various errors and inconsistencies. Now, all foreign names are simply tagged as **n----s**. The tagging of foreign names has become more intricate, especially when dealing with multi-word foreign names. For example, with a sequence of foreign words which form the name of a single entity, the first word is tagged as a proper noun (**n----s**). The rest of the words are tagged as foreign words (**e**) unless they are proper nouns in their own right[1]. The implementation of particular rules and exceptions caused a few difficulties which have yet to be dealt with. More on this in Section 6.2.

[1]Special situations which have their own rules include sports teams (e.g. *Golden n----s State e Warriors n----s.*) and car makes (e.g. *Ford n----s Focus n----s*)

3.2 New Tags in the Tagset

i) **Web Addresses**

- URLs and e-mail addresses are now tagged as **v**

No functionality was present in IceNLP to identify URLs and e-mail addresses so they were tagged as whatever seemed most appropriate to taggers (which was usually nonsensical). Now, they have their own unique tag which IceParser treats as a noun.

ii) **Punctuation**

- Punctuation marks now have their own tag category, **p**
- Commas are now tagged as **pk**

Punctuation marks got an overhaul in the tagset update. The "tag" given to punctuation marks was the punctuation mark itself, but now they have their own category. This is mostly irrelevant to IceParser as punctuation is generally not a part of a phrase. An exception to this are commas as they are very often used when listing a sequence of nouns or adjectives. Therefore, comma identification had to be updated.

iii) **Nouns**

- Shortened nouns now have their own tag category, **k**

Because of certain specific situations where shortened nouns have unique behaviour, they were given their own category.

- Abbreviated nouns are now tagged as **ks**

All abbreviations were formerly treated as multi-word adverb phrases in most cases. Now, that only applies to abbreviations where each word in a multi-word phrase is shortened to a single letter. An example of this is the adverb phrase *að minnsta kosti* (e. *at least*) which is always abbreviated as *a.m.k.*. Other abbreviations are mostly cases where a single noun, often a title or profession, is shortened. An example of this is when *lögreglustjóri* (e. *chief of police*) is shortened to *lögr.stj.* These abbreviations are now given their own tag which IceParser treats as a noun.

- Split nouns are now tagged as **kt**

A special phenomenon can occur in Icelandic when referring to two words that end with the same suffix. In Icelandic, one could, for example, refer to Friday and Saturday as *föstu- og laugardagur* (e. *Fri- and Saturday*). Note the hyphen where the suffix is removed from the former word. As these word fragments ending with a hyphen are unique scenarios in Icelandic, it was decided to give them their own tag. Using the previous example, the phrase *föstu- og laugardagur* is tagged as *föstu- kt og c laugardagur nken* with the new version of the tagset.

3.3 Tagset-Independent Changes

i) **Verbs**

- *Be*-verbs moved into a new transducer

As mentioned in Section 2.2, *be*-verbs need to be distinguished from other verbs. The verb "to be" does not have a special tag in the Icelandic tagset, but has a different role than other verbs. For example, a predicative complement follows *be*-verbs, whereas an object follows most other verbs. To successfully distinguish the two, it was necessary to split verb phrase annotations into separate transducers. In IceParser's pipeline, the new transducer which identifies *be*-verb phrases is run before the transducers which identifies other verb phrases, i.e. IceParser identifies all *be*-verb phrases and then identifies other verb phrases.

- Past participle verb identification functionality extended

IceParser formerly caught all past participles that were not parts of phrases and marked them as complements. This meant that many non-complement past participles were incorrectly marked as complements. Now, that functionality has been removed and replaced with more extensive functionality for identifying complements. This new functionality is currently incomplete and requires more robustness in particular situations. More on this in Chapter 6.2.

ii) **Foreign Words**

- Foreign name functionality moved into the noun transducer

Because of IceNLP's haphazard functionality regarding foreign names, it was kept in its own transducer. Now, foreign name functionality is better standardized so it doesn't need to be contained in its own file. Because foreign names are near exclusively treated as nouns, we decided that it was better to have that functionality in the same .flex file as nouns.

iii) **Qualifiers**

- Adjusted qualifier functionality within preposition phrases

In Icelandic, qualifiers are exclusively in the genitive case. IceParser uses this as the primary pattern of identifying qualifiers. IceParser formerly had a fairly naive implementation of this which caught non-qualifier genitive nouns, usually nouns which were genitive because of genitive prepositions. This implementation was tweaked which significantly reduced these wrong identifications, but there remain particular situations that aren't correctly matched. These are mostly when qualifiers appear within genitive prepositional phrases. These incorrect matches are further compounded when the nouns in question are foreign names. More on this in Section 6.2.

iv) **Temporal Expressions**

- Adjusted functionality of temporal expressions

IceParser formerly had very simplistic temporal expression functionality which wrongly identified many non-temporal expressions. This functionality has been removed and replaced by more precise temporal expression identification. As with the complements mentioned above, this new identification method requires more robustness to be considered complete. More on this in Section 6.2.

Chapter 4

Evaluation

4.1 Methods

A systematic approach was taken to find consistent errors in IceParser's functionality. To help with this, a 5000 sentence set extracted from a corpus was provided by Miðeind ehf. which could be run through IceParser at any time to measure its performance. Information about the contents of Miðeind's corpus can be found in [18].

It is worth mentioning that Miðeind's corpus consists of text from news articles and headlines. These texts come from various different sources which cover a large variety of topics. News websites are scraped on a daily basis, constantly increasing the size of the corpus.

This corpus was split into a 500 sentence set referred to as the test set and a 4500 sentence set referred to as the development set. Both sets were run through a state-of-the-art tagger built on ELECTRA-Base [19], a method for self-supervised language representation learning developed by Google Research.

This tagger, currently unnamed, is being worked on as part of the Language Technology Programme for Icelandic 2019-2023 [20]. The ELECTRA-Base tagger was trained on data from the Icelandic Gigaword Corpus or (IGC) [21], an Icelandic corpus containing over 1.5 billion words, and fine-tuned with the MIM-GOLD corpus [10], a million word tagged corpus whose tags has been manually corrected and functions as the gold standard for Icelandic POS tagging.

The test set was first run through IceParser and then manually corrected to create our own gold standard of what IceParser's results should be when parsing the test set. By running the test set through IceParser and then comparing the results with the gold standard, IceParser's performance could be evaluated.

The development set was also run through IceParser and its output analyzed in order to find errors. Errors were investigated by hand and identifications of error patterns were attempted. Some errors are a result of an oddly worded or wrongly ordered sentence, or even a sentence which contains misspellings. These errors were ignored in general as they either stem from errors outside IceParser or a unique situation that should not be accounted for.

Some errors happen due to the particular difficulty of the sentence being parsed. These errors are usually found in long sentences where particular word order cause unique exceptions. These errors were ignored as modifying IceParser to fix an error with as few as one occurrence can easily cause a snowball effect where fixing one error causes more errors to pop up. Other errors derive from patterns which IceParser either

fails to match or wrongly matches. These patterns of errors are the main focus of this part of the project.

Any consistent, systematic errors that have been identified have been compiled and categorized and are discussed in detail in Chapter 6.

4.2 Scoring

In 2020, Miðeind ehf. developed a test for different parsing schemas called ParsingTestPipe[1]. It measures the precision, recall, and F-measure of a parser by generating an automatically parsed version of tagged text using the parser being tested. It passes this parsed text and a hand-annotated version (gold standard) of the same text to Evalb [22], a bracket scoring program which measures the similarity between the two texts.

The F-measure [23] is a mathematical formula which gives a generally accepted percentage of accuracy. To calculate the F-measure[2], one must also calculate the precision and recall. The formulae for the three are the following:

$$Pr = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{4.1}$$

$$Re = \frac{True\ Positives}{True\ Positives + False\ Negatives} \tag{4.2}$$

$$F_1 = \frac{2 * Pr * Re}{Pr + Re} \tag{4.3}$$

where *True Positives* refer to the amount of phrases and functions that are correctly identified, *False Positives* refer to the amount of phrases and functions identified by IceParser that aren't in the gold standard, and *False Negatives* refer to the amount of phrases and functions in the gold standard that IceParser doesn't identify.

While the F-measure is a complete accuracy measurement in its own right, precision and recall are still highly relevant themselves for this project. More specifically, precision represents the proportion of phrases and functions in the automatically parsed text that are correct and recall represents the proportion of phrases and functions in the hand-annotated text that appear correctly in the automatically parsed version.

IceParser was administered three separate versions of tests with ParsingTestPipe: one which measured IceParser's phrase identification capabilities, one which measured IceParser's function identification capabilities, and one which measured both simultaneously.

The results of the test show that the total F-measure is 95.29% for phrases and is 79.48% for functions. When adjusted for the total number of phrases and functions, the total F-measure for the test set is 92.17%. Full results for phrases can be seen in Table 4.1 and for functions in Table 4.2 in which they are ordered by the amounts of each phrase or function found in the test set, placing the most common phrase and function type at the top of the corresponding table.

[1] `https://github.com/mideind/ParsingTestPipe`

[2] The abbreviation for the F-measure is F_1. This is to specify that the F-measure being used is the traditional or balanced F-measure. This particular measure is also known as the harmonic mean of precision and recall.

Some precision values are empty. These are cases where IceParser doesn't identify a single phrase or function of that type, either rightfully or wrongfully. The value is left blank as putting the values into Formula 4.1 would result in a scenario necessitating a division by zero. The F-measures in these situations default to 0%.

Phr. type	Pr (%)	Re (%)	F_1 (%)	Amount
NP	96.15	95.75	95.95	3133
PP	91.85	92.08	91.96	1162
VP	99.38	99.38	99.38	650
AP	96.60	96.77	96.68	558
AdvP	96.69	92.38	94.49	538
VPb	98.75	99.75	99.25	396
CP	100.00	98.48	99.23	395
SCP	100.00	98.51	99.25	268
VPi	100.00	99.21	99.60	254
VPp	97.58	100.00	99.78	161
NPs	51.79	67.44	58.59	129
MWE_CP	89.47	98.84	93.92	86
MWE_PP	69.09	95.00	80.00	40
MWE_AdvP	76.92	90.91	83.33	33
APs	89.47	89.47	89.47	19
InjP	100.00	100.00	100.00	4
VPg	100.00	100.00	100.00	3
FRWs	50.00	100.00	66.67	2
FRW	25.00	100.00	40.00	1
MWE_AP	100.00	100.00	100.00	1
All	95.07	95.52	95.29	7833

Table 4.1: Test results of individual phrase types

Func. type	Pr (%)	Re (%)	F_1 (%)	Amount
SUBJ>	92.87	88.85	90.82	601
OBJ<	87.39	80.21	83.65	389
QUAL	83.56	78.80	81.11	316
COMP<	79.51	68.20	73.42	239
SUBJ<	80.42	77.16	78.76	197
SUBJ	25.00	68.97	36.70	58
COMP>	82.00	93.18	87.23	44
TIMEX	61.54	50.00	55.17	32
IOBJ<	77.78	53.85	61.74	26
COMP	25.00	12.00	16.22	25
OBJ>	75.00	46.15	57.14	13
OBJNOM<	-	0.00	0.00	4
OBJAP<	100.00	33.33	50.00	3
OBJ	-	0.00	0.00	2
OBJAP	-	0.00	0.00	2
OBJNOM>	-	0.00	0.00	2
OBJAP>	100.00	100.00	100.00	1
OBJNOM	-	0.00	0.00	1
All	80.63	78.36	79.48	1955

Table 4.2: Test results of individual function types

4.3 Comparisons

For the sake of measuring the changes in accuracy, these new results were compared to those retrieved in [4], the introductory paper of IceParser. Before this project, these were the only accuracy measurements available for IceParser. For the comparison, the two scores for the overall F-measures for phrases and functions are most pertinent: 96.7% and 84.3%, respectively.

This equates to a 1.41 percentage point drop in performance for phrases and a 4.82 point drop in performance for functions.

While these statistics give an insight into the changes in performance, they cannot be considered to be particularly accurate because of the different scenarios in which they were measured, especially in regard to their test sets.

The test set which gave IceParser its original measurements was a set of 509 sentences randomly selected from the IFD corpus [5], a corpus made up of text from various Icelandic books. The test set provided by Miðeind used in this project, however, is made up of 500 sentences randomly chosen from what is mostly news articles and headlines as can be seen in [18].

The difference in sentences from either set are plentiful. For example, sentences in news articles tend to have far more foreign names, include uncommon elements like sports scores, and are longer on average. The gold standard for the IFD test set has 8281 individual words or punctuation, known as tokens, 6760 phrases and 1799 functions. The gold standard for Miðeind's test set contains 9514 tokens, 7833 phrases and 1955 functions. This translates to the original gold standard having an average of 16.27 tokens per sentence while the new gold standard has 19.03 tokens per sentence.

The new version of IceParser was run through the original test set which gives a more accurate comparison of the two versions. The original test is tagged according to the Icelandic tagset as it was in 2007, but this is not problematic as IceParser still has the functionality to handle older tagset versions as mentioned in Chapter 1. In this test, IceParser got a total F-measure of 96.34% for phrases and 83.09% for functions, which equates to a 0.35 point and 1.21 point drops in performance between versions, respectively. This accuracy seems to indicate that the new test set is more complex than the old one. Considering the origin of both test sets, one may even suppose that average sentences from news articles are more difficult to parse than those from literature. The full results of the original test set as well the difference in performance of the original and new versions of IceParser can be seen in Tables 4.3 and 4.4.

One thing to note with Tables 4.3 and 4.4 is that the results presented in [4] are formatted slightly differently than our results so the format of our results was altered to match. These differences include: all the different verb phrases being amalgamated into one phrase type (**VPx**), all the different multi-word expression phrases being amalgamated into one phrase type (**MWE**), and accuracy measurements being rounded to the nearest tenth instead of the nearest hundredth.

	F_1 (%)			
Phr. type	Orig.	New	Diff.	Amount
NP	96.8	96.8	0.0	2544
VPx	99.2	98.5	-0.7	1306
PP	96.7	95.9	-0.8	878
AdvP	91.8	91.3	-0.5	556
AP	95.1	95.1	0.0	545
CP	100.0	100.0	0.0	386
SCP	99.6	99.6	0.0	229
MWE	96.9	94.7	-2.2	169
NPs	80.4	80.4	0.0	100
APs	87.0	87.0	0.0	33
InjP	100.0	100.0	0.0	14
All	96.7	96.3	-0.4	6760

Table 4.3: Phrase F-measures of both IceParser versions and their differences.

	F_1 (%)			
Func. type	Orig.	New	Diff.	Amount
SUBJ>	92.7	93.7	+1.0	545
OBJ<	90.2	89.8	-0.4	354
COMP<	75.1	69.2	-5.9	228
SUBJ<	83.7	83.3	-0.4	222
QUAL	87.7	88.0	+0.3	188
SUBJ	68.2	67.3	-0.9	85
COMP	56.9	22.2	-34.7	50
TIMEX	74.7	60.0	-14.7	48
COMP>	91.3	76.4	-14.9	23
IOBJ<	73.3	80.0	+6.7	16
OBJ>	43.5	41.7	-1.8	14
OBJNOM<	30.8	0.0	-30.8	11
OBJAP<	75.0	66.7	-8.3	8
OBJ	0.0	0.0	0.0	4
OBJAP>	71.4	57.1	-14.3	3
All	84.3	83.1	-1.2	1799

Table 4.4: Function F-measures of both IceParser versions and their differences.

Chapter 5

Discussion

Looking at the tables in Chapter 4, some notable results are worth discussing, in regard to both high and low accuracies. What makes said results notable and why it occurs is the main topic of discussion.

Not all results are discussed. Any particularly low or high F-measures are explained, their reasons are mostly known. Any particularly large changes to IceParser's accuracies as determined by the comparison test in Section 4.3 are also analyzed.

5.1 Phrases

The F-measure of NPs is quite low (see Table 4.3), especially considering the high F-measure of NP. This is mostly due to a known error regarding foreign names, this is discussed further in Section 6.2. Similarly, the F-measure of APs is noticeably lower than that of AP. But, this is purely because of the low amount of APs phrases in the test set. Only two APs errors occur and both of those are due to nested AP errors within APs. NPs phrases can also suffer from nested errors, more on this in Section 6.1.

The F-measures of the different verb phrases are all very high, the lowest of the five having an F-measure of 99.25%. Verb phrases are very easy to identify in general and have little room for error. Any errors usually appear when a multi-verb verb phrase is incorrectly identified as several one-verb verb phrases.

Some miscellaneous phrase types have very high F-measures. These include CP, SCP, and InjP. These phrase types exclusively contain function words[1], whose purpose is more grammatical than lexical. This purpose also means that function words generally belong to closed classes, i.e. word classes which rarely have new words added to them. This makes them particularly simple to parse as these words have only one form, are unaffected by other words around them, and have a finite variety of words. Prepositions and infinitive particles are also function words but the phrases they belong to don't contain them exclusively. Preposition phrases include the preposition's object and infinitive particles are a part of the verb phrase.

Multi-word phrases are inconsistent as it is hit-or-miss whether IceParser will correctly identify them or not. A list of multi-word expressions is contained in IceParser which is the method used to identify them. If the expression is in the list, the MWE phrase is identified, otherwise it isn't. This causes lower recall due to unlisted expressions. Lower precision is caused by identifying MWE phrases that aren't there when

[1] In Icelandic, they are called *kerfisorð*.

certain words appear in the order as defined in the list of multi-word expressions. It is also important to note that MWE errors always cause different errors to pop up, most commonly with adverb phrases.

5.2 Functions

Common function errors come in several different varieties. Wrong or missing directions, function types being misidentified as a different function type, functions not capturing everything that it should, etc. These are not necessarily consistent, systematic errors, but are rather dependent on the structure of the sentence where certain situations cause these errors. Another thing to consider are the consequential errors from phrases. Because functions are parsed after phrases, any phrase errors become function errors down the line. More on this in Section 6.1.

The most common function type, the subject, has a very good F-measure (see Table 4.4). This is due to how Icelandic works. Icelandic is an SVO language, meaning that subjects usually appear at the start of sentences. They also usually appear in the nominative case, making them easy to identify and difficult to confuse with other function types. Non-standard variations of subjects (i.e. SUBJ and SUBJ<) are not as uniquely identifiable and, therefore, have lower accuracies.

Complements have lower accuracies for all three variations. COMP> are commonly confused with SUBJ> as both appear in the nominative case to the left of a verb. COMP has suffered the biggest reduction in accuracy according to Table 4.4. This is very likely due to the removal of a catch-all identifier which was a common source of directionless complements, this is discussed further in Section 6.2.

Nominative object identification appears to have completely disappeared in IceParser's newest version as not one instance was identified in either test set. This anomaly is not currently understood, but there is some consistency in these errors. More on that in Section 6.2.

Temporal expressions also suffer a reduction in accuracy. As mentioned in Section 3.3 change was implemented to the identification of temporal expressions. The reduction suggests that the robustness of this change is insufficient and more is needed to counter the removed functionality. A discussion of what must be done and why it is difficult can be found in Section 6.2.

Chapter 6

Error Analysis

By inspecting the test results, it becomes clear where errors and dysfunctionality remain within IceParser. Some of these have been identified already and are listed in this chapter. Others currently have unknown causes and effects and a more comprehensive error analysis must be performed. For that purpose, the detailed test results in Chapter 4 can be used to pinpoint where these unknown errors likely originate.

Locating systematic errors is done by running the 4500 sentence dev set (defined in Section 4.1) through IceParser at any point during modifications and looking through the results by hand, searching for any consistently occurring mistakes. When similar mistakes happen in similar situations, this is considered likely to be an area of functionality where IceParser is failing to parse a phrase or function correctly.

Debugging is then done via trial-and-error code modification where certain parts of code are changed and the results of IceParser checked. This process requires an eye for detail to determine what specific code to change. To find out where changes are necessary, one must figure out, for example, whether an error is a consequence of some other functionality, whether changes are necessary upstream or downstream in the pipeline (i.e. in an earlier or later transducer in the order of transducers, respectively), and how to implement the necessary change without compromising previous functionality. To give an example, if any changes made in one transducer fails to have any effect on the current error, this is a strong sign that the error occurs in an earlier transducer. Follow-up checks on other IceParser functionality is then recommended and it is therefore good to have in mind while debugging what downstream functionality may be affected.

One important aspect of error analysis to consider has to do with nested phrases, i.e. phrases within other phrases. If the inner phrase is wrong, then Evalb considers both the inner and outer phrase to be incorrect even if the outer phrase is otherwise correct. This means that a single error can be interpreted as multiple errors in certain situations.

Persistent errors that are still present in IceParser after modifications can be put into one of two categories: Errors that existed before modifications and haven't yet been addressed, and errors that came up as a result of the modifications.

6.1 Previous Errors

i) **Directionless complements after adverb phrases**

When a sentence lacks a main verb but has a complement, IceParser still has the capability to identify the complement as the transducer which handles complements has built-in functionality to do so. Sentences of this type are often found in news headlines, e.g. *Kosningum frestað* (e. *Elections postponed*). Naturally, the same complement should be identified when an adverb phrase is added into the sentence, e.g. *Kosningum ekki frestað* (e. *Elections not postponed*), but this is not the case with IceParser. For unknown reasons, an adverb phrase disrupts the pattern matching abilities of IceParser, even when explicitly accounted for. It is believed that this is a result of a quirk in IceParser's code, but the cause has yet to be identified.

ii) **Adverbs as part of adjective phrases**

Adverb phrases can often be used to modify adjective phrases by appearing in front of them. In these situations, the adverb is considered a part of the adjective phrase. For example, the phrase *mjög gott* (e. *very good*) is parsed as *[AP [AdvP mjög aa] gott lhensf]* . However, adverb phrases can appear before adjective phrases without it functioning as a modifier and, therefore, should not be part of the adjective phrase. For example, the phrase *einungis ríkustu* (e. *only the richest*) should be parsed as *[AdvP einungis aa] [AP ríkustu lhfnve]* . IceParser cannot distinguish between these and considers all adverbs to be modifiers if they appear before adjectives. Because adverbs are generally tagged with a single tag (**aa**), the solution may be to create a list of adverbs that would be modifiers when appearing in front of adjective phrases. However, this is only speculation and more investigation is required.

iii) **Function errors as a consequence**

Due to IceParser identifying phrases from the tagged text and identifying syntactic functions from said phrases, any errors that come up in phrase identification result in errors in function identification. This is a unique type of problem whose preferred solution is to indirectly fix it by reducing phrase identification errors. This type of error is important to mention as it suggests that function identification accuracy cannot be higher than phrase identification accuracy.

An example of this error can be seen with the example of the previous adjective phrase error as a subject function. If the adjective phrase *einungis ríkustu* (e. *only the richest*) is a subject, then the correct procedure is for the subject transducer to contain the adjective phrase, i.e. *[AdvP Einungis aa] {*SUBJ> [AP ríkustu lhfnve] }*. If the adjective phrase is incorrectly parsed as in the error shown, the SUBJ function would indirectly contain the adverb phrase, i.e. *{*SUBJ> [AP [AdvP Einungis aa] ríkustu lhfnve] }*. This would make the subject function incorrect even though it is performing exactly as it should, identifying the adjective phrase as the subject.

iv) **Phrase errors as a consequence**

Preposition phrases, alongside NPs and APs phrases, are unique among phrases as they are the only types of phrases that are necessarily nested, e.g. preposition phrases always include the preposition and the noun phrase which functions as the preposition's object. As mentioned, errors in nested phrases result in errors in both the inner and outer phrases. This means that there is always a risk of these phrase types being classified as incorrect despite their phrase identification functioning correctly. Again, the preferred solution is to indirectly fix it by reducing the error rate of phrase types which can appear within these phrase types, like noun phrases and adjective phrases. NPs and APs are particularly susceptible to this kind of error as they contain at least two nested phrases: two noun phrases and two adjective phrases, respectively, often including a conjunction phrase. This increases the chances of nested errors even more.

An example of this could be seen in the (now fixed) error where qualifiers wrongfully appeared inside preposition phrases. Take the simple preposition phrase *til hans* (e. *to him*). The preposition phrase would contain the preposition and the noun phrase which acts as the preposition's object, i.e. *[PP til af [NP hans fpkee]]*. If the noun phrase is then wrongly identified as a qualifier, i.e. *[PP til af {*QUAL [NP hans fpkee] }]*, the preposition phrase is marked as wrong even though it performed exactly as hoped, capturing the preposition and the noun phrase.

6.2 New Errors

i) **Past participles as complements**

As mentioned in Section 3.3, all past participles, that were not identified as belonging to a *be*-verb (either annotated as COMP> or COMP<) were marked as "standalone" complements (annotated as COMP). This was removed to avoid wrongfully marking non-complements as complements. But this function did correctly mark complements as such that were otherwise being missed by IceParser. To counter this, other means of rightfully identifying these complements were implemented instead. However, there remain particular scenarios where these complements are not identified by IceParser, especially when complements are isolated from phrases which could help identify complements.

For example, in the sentence *Er maðurinn í hattinum ekki farinn?* (e. *Is the man in the hat not gone?*) the past participle *farinn* (e. *gone*) is a complement and the main verb *er* (e. *is*) proves that, but in between those are a noun phrase (*maðurinn* (e. *the man*)), a preposition phrase (*í hattinum* (e. *in the hat*)), and an adverb phrase (*ekki* (e. *not*)). The variety of phrase sequences that can isolate a complement like this is vast and IceParser would need more robust pattern matching capabilities when identifying complements.

It is worth mentioning that the complements marked by the catch-all functionality in these situations would be marked as errors as IceParser would identify them as directionless, whereas the correct answer according to the gold standard would be a complement pointing towards the verb. This error is, therefore, not technically a new error, but a different version of an existing error.

ii) **Foreign names in multi-noun phrases**

When grouping together noun phrases, a specific transducer identifies whether two or more nouns next to each other should be grouped or kept separate. It does this by looking at the nouns' attributes, matching cases being the best clue that they should be grouped. As mentioned in Section 3.1, foreign names are now tagged with **n----s** which intentionally leaves out any possible attributes. This makes it difficult for IceParser to determine whether a foreign name should be grouped with adjacent noun phrases or not. The current method is one that always assumes that these groupings should occur as these scenarios are rare enough that incorrect identifications of multi-noun phrases are few and far between, but this does mean that multi-noun phrases with foreign names do appear when they're not supposed to.

For example, take the sentence *Hann tók boltann og Tom grét* (e. *He took the ball and Tom cried*) where *boltann* (e. *the ball*) and *Tom* are separate noun phrases with separate roles. Because *Tom* would be identified as foreign name, IceParser wrongly identifies *boltann og Tom* (e. *the ball and Tom*) as a multi-noun phrase. A better method would be to give IceParser more robust pattern matching when dealing with these scenarios, though that does require a better insight into what specific scenarios must be accounted for.

iii) **Missing temporal functions**

Similarly to the previously mentioned complements, IceParser had a function which marked all non-assigned (pronoun, noun) pairs or (numeral, noun) pairs as temporal functions. While counter-intuitive, this did manage to catch unidentified temporal functions such as *einn dag* (e. *one day*) or *allt sumar* (e. *all summer*). Much like the complements, this was misidentifying a lot of phrases as temporal expressions. This catch-all functionality was therefore removed to be replaced by a more robust pattern matching schema. It proved to be a challenge to account for all possible temporal functions, mostly because of the sheer amount of unique nouns in possible (pronoun/numeral, noun) pairs.

The list of necessary nouns to look for includes, but is not limited to: all months and weekdays (and their declensions), different lengths of time (i.e. weeks, years), special events (i.e. Christmas, Easter), and non-temporal (pronoun,noun) pairs used to denote contextual lengths of time (i.e. *all journey*, *that walk*). An additional step to all of this is that the pattern matching of temporal expressions is the first in IceParser's syntactic function pipeline which means that extra care has to be taken to not match any phrases which should be identified as other functions (e.g. subjects, complements). This is is done by only matching certain declensions as temporal functions. All of this means that temporal function pattern matching is incomplete, resulting in some temporal functions being unidentified.

iv) **Missing nominative objects**

For unknown reasons, the identification of nominative objects (OBJNOM<, OBJNOM>, OBJNOM) has seemingly disappeared from the newest version of IceParser. Though rare, several instances of nominative objects exist in both test sets used in this project, but not one is identified by IceParser. This is especially curious as no changes have been made to the functionality of nominative objects, either in the tagset revision or IceParser modifications.

While unidentified, some patterns appear consistently where nominative objects should appear. For example, many instances are wrongly identified as directionless subjects (SUBJ), suggesting that they are being identified by a later transducer which tags unassigned nominative phrases as SUBJ. This implies that some modification to IceParser is causing nominative objects to slip through other transducers unidentified until it reaches this late catch-all identifier.

Chapter 7

Conclusion

In this project, we have presented modifications to IceParser, a shallow parsing tool for Icelandic. These modifications were necessary as the Icelandic tagset, which provides crucial information to IceParser, went through a major revision. Along with these necessary modifications, additional improvements were made to IceParser to further improve its performance.

We covered the biggest modifications in detail and put each into one of three categories depending on what factor prompted the modification: Changes to the tagset, additions to the tagset, and tagset-independent changes. We also performed a preliminary error analysis of persistent errors known to remain in IceParser, putting each into one of two categories: Those that existed before the modifications, and those that didn't.

An evaluation was carried out to measure the accuracy of IceParser after the modifications. For this evaluation, a set of text from news articles and headlines was used. The evaluation shows that IceParser obtains 95.29 and 79.48 F_1-score for phrases and syntactic functions, respectively. These results suggest that performance has dropped by 1.36 percentage points for phrases and 4.06 points for functions from IceParser's original accuracy scores [4]. Because the accuracy measurements were made using two very different corpora, however, we consider them to be incomparable to each other.

A secondary evaluation was administered for the sake of comparison to IceParser's original performance. With this evaluation, IceParser's F_1-score was 96.34 and 83.09 for phrases and functions, respectively. These results are directly comparable to IceParser's original scores as the same test set was used for both, a set which uses text from books. By comparing F_1-scores for both versions of IceParser, we showed that performance between the two versions drops by 0.39 percentage points for phrases and 1.21 points for functions. These results suggest two things: First, that IceParser's overall performance drop is not as significant as implied by the first test and, second, that news articles contain text that is more difficult to parse than that of literature.

For the sake of future modifications, usage of balanced test sets with text from multiple sources (and, more importantly, multiple types of sources) is highly recommended. Because of the imbalanced (i.e. single source type) test sets used in both tests, we can't, in good conscience, call these test results a fully accurate performance metric for IceParser. They do, however, still give good insight into IceParser's performance and we do consider the results to be a good enough approximation of the true performance.

The nature of this project means that the modifications presented in this project are by no means the last to be made to IceParser. Future work will be done to continue increasing the parsing accuracy, either by improving IceParser itself or by improving other constituent parts of IceNLP, most likely both. This is the purpose of the preliminary error analysis. By going into detail regarding precision and recall to individual types of phrases and functions, we pinpoint the source of errors and highlight areas where new modifications are most necessary.

Bibliography

[1] A. Kornai, "Digital Language Death", *PLOS ONE*, vol. 8, pp. 1–11, Oct. 2013. DOI: 10.1371/journal.pone.0077056. [Online]. Available: https://doi.org/10.1371/journal.pone.0077056.

[2] E. Rögnvaldsson, K. M. Jóhannsdóttir, S. Helgadóttir, and S. Steingrímsson, "The Icelandic language in the digital age", *White Paper Series. META-NET*, 2012.

[3] H. Loftsson and E. Rögnvaldsson, "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, 2007.

[4] Loftsson, Hrafn and Rögnvaldsson, Eiríkur, "IceParser: An incremental finite-state parser for Icelandic", in *Proceedings of the 16th Nordic Conference of Computational Linguistics (NoDaLiDa 2007)*, Tartu, Estonia, 2007, pp. 128–135.

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

[6] A. K. Ingason, S. Helgadóttir, H. Loftsson, and E. Rögnvaldsson, "A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI)", in *Advances in Natural Language Processing*, B. Nordström and A. Ranta, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 205–216, ISBN: 978-3-540-85287-2.

[7] H. Loftsson, "Tagging Icelandic text: A linguistic rule-based approach", *Nordic Journal of Linguistics*, vol. 31, no. 1, p. 47, 2008.

[8] H. Loftsson and E. Rögnvaldsson, "Linguistic richness and technical aspects of an incremental finite-state parser", in *Proceedings of "Partial Parsing 2008", workshop at the 6th International Conference on Language Resources and Evaluation, LREC 2008*, Marrakech, Morocco, 2008.

[9] E. Rögnvaldsson, *CLARIN-IS*, https://clarin.is/, (Accessed April 2021), 2018.

[10] S. Barkarson, E. F. Sigurðsson, E. Rögnvaldsson, H. Hafsteinsdóttir, H. Loftsson, S. Steingrímsson, and Þ. D. Andrésdóttir, *MIM-GOLD 20.05*, CLARIN-IS, 2020. [Online]. Available: http://hdl.handle.net/20.500.12537/39.

[11] S. Abney, "Part-of-Speech Tagging and Partial Parsing", in *Corpus-Based Methods in Language and Speech Processing*. Dordrecht: Springer Netherlands, 1997, pp. 118–136, ISBN: 978-94-017-1183-8. DOI: 10.1007/978-94-017-1183-8_4. [Online]. Available: https://doi.org/10.1007/978-94-017-1183-8_4.

[12] H. Loftsson and R. Östling, "Tagging a morphologically complex language using an averaged perceptron tagger: The case of icelandic", in *19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, 2013, pp. 105–119.

[13] B. Megyesi and S. Rydin, "Towards a Finite-State Parser for Swedish", in *Proceedings of the 12th Nordic Conference of Computational Linguistics (NODALIDA 1999)*, 2000, pp. 115–123.

[14] H. Loftsson and E. Rögnvaldsson, "A shallow syntactic annotation scheme for Icelandic text", Department of Computer Science, Reykjavík University, Tech. Rep. RUTR-SSE06004, 2006.

[15] G. Klein, S. Rowe, and R. Décamp, *JFlex - The Fast Scanner Generator*, `https://jflex.de/`, (Accessed February 2021), 1998.

[16] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[17] M. Fowler, *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 2018.

[18] V. Þorsteinsson, *Greynir - Málgreining fyrir íslensku*, `https://greynir.is/stats`, (Accessed April 2021), 2017.

[19] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators", in *ICLR*, 2020. [Online]. Available: `https://openreview.net/pdf?id=r1xMH1BtvB`.

[20] A. B. Nikulásdóttir, J. Guðnason, A. K. Ingason, H. Loftsson, E. Rögnvaldsson, E. F. Sigurðsson, and S. Steingrímsson, "Language Technology Programme for Icelandic 2019-2023", in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, Marseille, France, 2020, pp. 3414–3422.

[21] S. Steingrímsson, S. Helgadóttir, E. Rögnvaldsson, S. Barkarson, and J. Guðnason, "Risamálheild: A Very Large Icelandic Text Corpus", in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, 2018.

[22] S. Sekine and M. J. Collins, *Evalb - bracket scoring program*, `https://nlp.cs.nyu.edu/evalb/`, (Accessed April 2021), 1997.

[23] N. Chinchor, "MUC-4 Evaluation Metrics", in *Proceedings of the 4th Conference on Message Understanding*, ser. MUC4 '92, McLean, Virginia: Association for Computational Linguistics, 1992, pp. 22–29, ISBN: 1558602739. DOI: `10.3115/1072064.1072067`. [Online]. Available: `https://doi.org/10.3115/1072064.1072067`.