A Wide-Coverage Context-Free Grammar for Icelandic and an Accompanying Parsing System

Vilhjálmur Þorsteinsson, Hulda Óladóttir
Miðeind ehf.
Reykjavik
Iceland
vt@extrada.com, holado@gmail.com

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

Abstract

We present an open-source, wide-coverage context-free grammar (CFG) for Icelandic, and an accompanying parsing system. The grammar has over 5,600 nonterminals, 4,600 terminals and 19,000 productions in fully expanded form, with feature agreement constraints for case, gender, number and person. The parsing system consists of an enhanced Earley-based parser and a mechanism to select best-scoring parse trees from shared packed parse forests. Our parsing system is able to parse about 90% of all sentences in articles published on the main Icelandic news websites. Preliminary evaluation with *evalb* shows an F-measure of 71.90% on parsed sentences. Our system demonstrates that parsing a morphologically rich language using a wide-coverage CFG can be practical.

1 Introduction

A CFG consists of a set of production rules that recursively describe how the strings of the underlying language can be derived. A parser for a natural language CFG checks whether a sentence can be derived by the CFG, and if so, assigns a syntactic structure (one or more parse trees) to the sentence. Well-known general parsing algorithms for CFGs include the (bottom-up) CYK algorithm (Younger, 1967) and the (top-down) Earley algorithm (Earley, 1970).

Various textbooks on Natural Language Processing (NLP) contain toy CFGs (for English, in most cases, e.g. (Jurafsky and Martin, 2009; Ljunglöf and Wirén, 2010)), but very few papers in recent literature describe hand-crafted natural language CFGs and parsers for them (we discuss a couple of them in Section 2). The reason is that the development of wide-coverage hand-crafted CFGs has been viewed as a challenging and time-consuming task (Briscoe et al., 1987; Hindle, 1989; Kinyon and Prolo, 2002), and that common phenomena, like agreement and inflection, have been considered complicated to describe using a CFG (Ljunglöf and Wirén, 2010). The processor and memory requirements of fully general parsing algorithms that can handle ambiguous grammars, being of worst-case cubic order (Younger, 1967; Scott, 2008), have also been seen as prohibitive. Beyond parsing, hand-crafted CFGs can be used *inter alia* to check and correct grammar in text (see Section 3).

Several treebanks have been developed for various languages during the past 25 years or so, e.g. (Marcus et al., 1993; Brants et al., 2004; Rögnvaldsson et al., 2012). Thus, instead of developing a CFG for a given language and a parser for the grammar, the more common approach has been to induce a probabilistic CFG (PCFG) from a treebank, and to train a probabilistic parser on the PCFG (Cahill, 2008). However, creating a high-quality treebank is a labor-intensive task, and, indeed, in many aspects similar to the process of manually crafting a grammar (Xia, 2001). Moreover, attempts to apply probabilistic parsing to morphologically rich languages (MRLs) have in many cases yielded unsatisfactory results, as complex word structure and flexible word order cause data sparseness in the probabilistic model (Tsarfaty et al., 2013).

In this paper, we present the development of an open-source, wide-coverage CFG for Icelandic, an MRL in the Germanic language family. We also present a parsing system, based on an enhanced Earley parser, that performs Part-of-Speech (PoS) tagging and parsing in a combined algorithm, and is able to cope with the large CFG and high levels of ambiguity associated with an MRL. It uses heuristics to return a single best scoring tree from the (often very large) packed parse forest for a sentence.

This paper is organized as follows: We discuss related work in Section 2 and the motivation for our work in Section 3. Tokenization and Out-of-Vocabulary words are discussed in Sections 4 and 5, respectively. We describe the development of the CFG in Section 6 and the parsing system in Section 7. The system is demonstrated in Section 8 and an evaluation is presented in Section 9. Finally, we conclude in Section 10.

2 Related Work

As discussed in Section 1, it is difficult to find published work from recent years describing the development of wide-coverage natural language CFGs and parsers for them. Here, we mention a couple of papers that we are aware of.

A CFG for English, developed over several years, is part of the GATE (General Architecture for Text Engineering) system, developed at the University of Sheffield (Gaizauskas et al., 2005). The CFG was specifically developed to complement a general purpose bottom-up chart parser called SUPPLE for feature-based CFGs, i.e. a CFG augmented with features in order to enforce agreement.

Abbas (2016) describes an extended Earley algorithm for parsing Urdu, an MRL. The parser uses a CFG extracted from a small treebank of 1,400 sentences (in comparison, the Penn Treebank (Marcus et al., 1993) contains 40,000 sentences). The small size of the treebank is, presumably, the main reason for not training a probabilistic parser. Moreover, the author mentions that it is hard to achieve good parsing results with probabilistic parsers for an MRL, without explicit encoding of linguistic information.

Before the work presented in this paper, no full parser existed for Icelandic. On the other hand, a shallow parser, *IceParser*, has been developed for the language (Loftsson and Rögnvaldsson, 2007). IceParser is an incremental finite-state parser, which annotates both constituent structure and syntactic functions, given PoS tagged input. The annotation scheme was designed in the project.

One Icelandic treebank, *The Icelandic Parsed Historical Corpus (IcePaHC)*, has been developed (Rögnvaldsson et al., 2012). It consists of one million words of historical texts, from the late 12^{th} century to the present, with about 10% being modern Icelandic. The annotation scheme follows the Penn Treebank style. IcePaHC has been used to train a probabilistic parser in a project aimed at improving the parsing accuracy of a related language, Faroese (Ingason et al., 2014).

3 Motivation

An initial primary goal of our project was to be able to extract information from text on Icelandic websites, such as associating person names with titles, and named entities with their definitions, irrespective of the grammatical forms that such associations may take in the text. As the project evolved, a secondary goal became to leverage the CFG and the parser to check and correct grammar, by adding specially annotated grammar rules and parser configurations for common error constructs. Finally, as our corpus of automatically parsed sentence trees grew to an order of millions, it became the basis of a follow-on project, beyond the scope of this paper, to train a deep neural network to parse Icelandic text.

4 Tokenization and Annotation

One of the catalysts for the project was the availability of the *Database of Modern Icelandic Inflection* (DMII) (Bjarnadóttir, 2012)[1]. DMII is a database that maps over 6 million lexical entries, 2.8 million unique word forms, to lemmas as well as PoS tags (word categories and morphological features).

The first phase of our project involved writing a tokenizer that uses data from the DMII to annotate each word token of a sentence with the set of its possible lemmas and corresponding PoS tag profile (possible PoS tags). In Icelandic, even common words such as *á* are highly ambiguous; *á* can mean *own* (verb), *female sheep* (noun), *river* (noun), or *on* (preposition), and can also occur as an adverb and an exclamation – our system associates it with 14 different PoS tags. This again means that the token for *á*, and indeed most word tokens, can match several different terminals in the CFG.

The tokenizer greedily recognizes certain multi-token spans, such as dates and adverbial multi-

[1] The DMII is available for download at `http://bin.arnastofnun.is/dmii/`.

word idioms, and coalesces them into single compound tokens. This makes the parsing stage more efficient and reduces ambiguity.

5 Out-of-Vocabulary Words

Compounding in Icelandic is very productive, and thus out-of-vocabulary (OOV) words, i.e. word forms that are not present in the DMII, are frequently encountered. We handle this by implementing a de-compounding algorithm, on top of a *Directed Acyclic Word Graph* comprising all word forms in the DMII, that maps each OOV word to a minimal sequence of prefixes followed by the longest possible suffix. This suffix is then used as a proxy for the compound word when looking up its PoS tags in the DMII.[2]

If the de-compounding algorithm is unable to make sense of a word, e.g. a foreign word, spelling error, or previously unseen named entity, its token is assumed to represent a neutral gender, singular noun that matches noun terminals in any of the four cases.

6 Context-Free Grammar

Once we had designed the mechanism for token-to-terminal matching (further described in Section 6.1), we proceeded to write out the nonterminals and productions of our CFG in an iterative fashion. Initially, a kernel of core nonterminals and productions for basic sentence structures and simple forms of noun, verb, prepositional and adverbial phrases was created.[3] This small CFG was then used to parse a starting reference set of typical texts selected from news articles. In each iteration, the most significant gaps in the CFG were identified and a round of improvements was applied. The set of reference texts was gradually expanded as the CFG coverage increased. This cycle is still ongoing, albeit of course with diminishing returns.

A web-based graphical user interface (GUI) was developed at an early stage in the project (see Section 8). The GUI provides an overview of news articles where sentences that the system fails to parse are clearly identified, and allows inspection of parse trees in graphical format. Having a visual way to identify problem areas and assess the system's performance facilitated the iterative cycle.

Our CFG for Icelandic is defined in a text file, presently about 5,800 lines including comments, written in a superset of Enhanced Backus-Naur Format (EBNF). This superset is our own implementation which facilitates automatic expansion of production rules (as discussed in Section 6.2). Apart from the CFG itself, the file contains annotation pragmas, such as priority scores for nonterminals (further discussed in Section 7.2). The total effort to date on the construction of the CFG is on the order of 2–3 man years.

6.1 Terminals

Our CFG allows three types of terminals:

- **Literal terminals**: Terminals of the form `"text"` match tokens with that text only (case-insensitive). This terminal form is, for example, used for various types of conjunctions.

- **Lemma terminals**: Terminals of the form `'lemma:category'_var1_var2...` match tokens having at least one PoS tag matching the indicated word category with the given lemma, optionally also agreeing with the specified variants (see Section 6.2). As an example, `'hafa:vb'_sg` (*hafa* being the infinitive of the verb *to have*) matches any singular form (`_sg`) of the verb, including *hef* (*[I] have*), *hafðir* (*[you] had*) and *hefur* (*[she] has*). This terminal form is, for example, used for auxiliary verbs in complex verb constructions.

- **Lookup terminals**: Terminals of the form `category_var1_var2...` match tokens having at least one PoS tag with the indicated word category that agrees with all of the specified variants, if any. As an example, `no_neut_sg_nom` matches any word token that has a PoS tag that identifies it as a noun (`no`), neutral (`_neut`), singular (`_sg`), nominative case (`_nom`). A word token such as *veður* (meaning *weather*) would match the terminal `no_neut_sg_nom`, but the same token also has a PoS tag indicating a verb (meaning *(he) wades*) and would thus also match the terminal `vb_sg_p3` that specifies a singular, third person verb.

[2] An example is *menntamálaráðherra*, (the minister of matters of education), composed of *mennta-mála-ráð-herra*, i.e. three prefixes and the suffix noun *herra*, from which the inflection of the compound word is derived.

[3] The general outlines of such a kernel would be similar for most languages in the Germanic family.

Feature	Variant	Values
Gender	`/gender`	`_masc, _fem, _neut`
Number	`/number`	`_sg, _pl`
Case	`/case`	`_nom, _acc, _dat, _gen`
Person	`/person`	`_p1, _p2, _p3`

Table 1: The major variants used in our CFG for Icelandic.

6.2 Variants and Feature Agreement

Describing nontrivial, potentially long-distance feature agreement in CFGs has been considered a challenge. In our grammar, we use automatic expansion of macro-like constructs, called *variants*, for this purpose. Variants are defined for the morphological features for which agreement is required within the productions of a nonterminal. They can be applied to both nonterminals and terminals. Table 1 shows the major variants used in our CFG for Icelandic.

Terminals with variants can only match word tokens that have at least one PoS tag that matches all of the variants. If a terminal has, e.g., an accusative case variant (`_acc`), it can only match word tokens that have a PoS tag indicating the accusative in the DMII.

As a simplified (and anglicized) example of how feature agreement is specified in the CFG, consider the following fragment:

```
NounPhrase ->
  Determiner/case/number/gender?
  Noun/case/number/gender
Determiner/case/number/gender ->
  det/case/number/gender
Noun/case/number/gender ->
  no/case/number/gender
```

Here, the nonterminal `NounPhrase` is defined as an optional `Determiner` having case, number and gender variants (`/case/number/gender`), followed by a mandatory `Noun` nonterminal, again with case, number and gender variants that agree in each expansion with the ones in the `Determiner`. The production for `Determiner` consists of a single terminal, `det/case/number/gender`, which matches word tokens having a PoS tag with the `det` category, inflected in accordance with the variants. The same applies to the production for `Noun`, which contains a single `no` terminal matching any noun that agrees with the features specified by the variants.

When the CFG is parsed, the variants are expanded as macros having each of their feature values in turn. The fragment above is expanded from three production rules to a total of 3 rules * 4 cases * 2 numbers * 3 genders = 72 expanded rules in the grammar. They include:

```
NounPhrase ->
    Determiner_nom_sg_masc? Noun_nom_sg_masc
  | Determiner_nom_sg_fem? Noun_nom_sg_fem
  | Determiner_nom_sg_neut? Noun_nom_sg_neut
...[21 generated productions omitted]...
Determiner_nom_sg_masc -> det_nom_sg_masc
Determiner_nom_sg_fem -> det_nom_sg_fem
Determiner_nom_sg_neut -> det_nom_sg_neut
...[21 generated productions omitted]...
Noun_nom_sg_masc -> no_nom_sg_masc
Noun_nom_sg_fem -> no_nom_sg_fem
Noun_nom_sg_neut -> no_nom_sg_neut
...[21 generated productions omitted]...
```

7 Parsing System

7.1 Earley-Based Parser

Parsing algorithms for natural language text can be evaluated on a number of criteria. They need to be able to parse all well-formed CFGs, including (heavily) ambiguous ones. A direct relationship between the produced parse forests and the original, unmodified CFG is a desirable feature. Last, but not least, the parsing performance in terms of time and space must be good enough for real-world applications.

The Earley (1970) algorithm handles all CFGs and, when extended from its original recognizer form to a full parser, returns parse trees that correspond directly to the original grammar (i.e. not requiring any *ex ante* transformation of the CFG). However, performance has been seen as a problem when parsing heavily ambiguous sentences, with both time and space requirements being worst-case cubic ($O(N^3)$) in the sentence length *N*.

Tomita (1986) described *Shared Packed Parse Forests* (SPPFs), a data structure that avoids redundancy when generating and storing parse trees for ambiguous token spans. Tomita's parser was a Generalized LR parser of worst-case unbounded polynomial order. Later, Scott (2008) and Scott and Johnstone (2010) presented an Earley-based parser which produces a binarized SPPF representation of all derivations of a sentence in worst-case cubic time. This parser meets all of the above mentioned criteria.

We implemented the Earley-Scott-Johnstone algorithm in C++ and found it to perform well enough for parsing natural language text according to our highly complex and ambiguous CFG for Icelandic.[4]

[4] Our system takes just over 1 second of wall-clock time

7.2 Disambiguation

The parser produces a binarized SPPF that includes all possible derivations of a sentence. A typical ambiguity factor is around 1.8, meaning that a sentence of N tokens typically has around 1.8^N different parse trees.

A nonterminal node in the SPPF may have more than one family of children if there are multiple derivations of that nonterminal for the same token span. Note that a single packed subtree may occur within multiple parse trees in a forest, by virtue of the packing mechanism.

Once the parse forest has been derived, bottom-up scoring heuristics are applied in a disambiguation step to find a single "best" parse tree. We begin the disambiguation process at the terminal (bottom) level of the SPPF, by assigning a score to each token-terminal match based on the probability of that match. For instance, the word *ekki* can be both an adverb (meaning *not*) and a noun (meaning *sob*, as in crying). The former meaning is much more common than the latter one, and thus a match of *ekki* to a noun terminal gets a relatively lower score, while a match of *ekki* to an adverb terminal gets a relatively higher score. The scores are heuristically defined and hand-tuned by us.

The scores assigned at the terminal level are summed up as they percolate upwards in the SPPF, and further adjusted at nonterminal nodes. In the CFG specification, scores can be assigned to nonterminals using pragmas. More common grammatical constructs have positive scores, while less common or exceptional ones have negative scores.

When the traversal encounters an ambiguous nonterminal node with multiple families of children, the accumulated scores of the families are compared. The family (subtree) with the highest score "wins" and is retained, while the other families are pruned from the tree. At the root (top) nonterminal, a single highest-scoring tree remains and is returned. It is at this point that the PoS tag for each word token is finally determined.

7.3 Attachment of Prepositions

There is, however, one exception to the general case described in the last subsection: The mechanism for attaching prepositional phrases (PPs) to verb or noun phrases. This well-known problem manifests itself in ambiguous sentences such as: *Ég las blaðið í gær* (*I read the paper yesterday / I read yesterday's paper*); should the PP *í gær* (*yesterday*) be attached to the verb *las* (*read*) or to the noun *blaðið* (*the paper*)? The problem is not perfectly solvable since the correct attachment may depend on semantics. We approximate a solution by augmenting our database of almost 7,300 verbs with a hand-crafted list of prepositions that are commonly attached to each verb. For example, with the verb *tala*, meaning *talk*, we associate the preposition *við/acc* (meaning *to/accusative*), indicating that the preposition *við*, when governing an object in the accusative case, is commonly attached to the verb *tala*.

Recall that our parser generates an SPPF representing all possible parse trees, including ones where PPs are attached to verbs and ones where they are attached to nouns. We want to selectively boost the score of subtrees where a PP (such as *við/acc*) is attached to a verb phrase (VP) with a matching verb (such as *tala*). This makes such subtrees more likely to "win" in the disambiguation and tree pruning process, vs. subtrees where the same PP is attached to a noun or to a non-matching verb.

In order for this to work correctly, we must partially unpack the parse forest. Specifically, PP nodes are unpacked (cloned) to become independent children of any ambiguous parent nonterminal nodes up to and including the enclosing VP node. This is done in a special top-down preprocessing pass before the main disambiguation scoring and pruning pass. During the scoring pass, we keep track of current VP scopes (such as the scope of *tala*) and assign scores to preposition terminal nodes (such as *við*) depending on whether the preposition matches an enclosing verb. Since the PP nodes have been unpacked, they can have different scores in different subtrees and can be independently pruned from the forest, which the SPPF structure would otherwise not allow.

8 Demonstration

Our system is open and available both as a public website[5] and as a GNU GPLv3-licensed Python package for NLP researchers and developers.[6]

to parse, disambiguate and process a typical 22-sentence news article from the web, on a quad-core Intel® i7 based GNU/Linux server.

[5] `https://greynir.is` (only in Icelandic).

[6] The source code for the CFG and the parsing system, written in Python 3 and C++, is available at `https://github.com/mideind/ReynirPackage`.

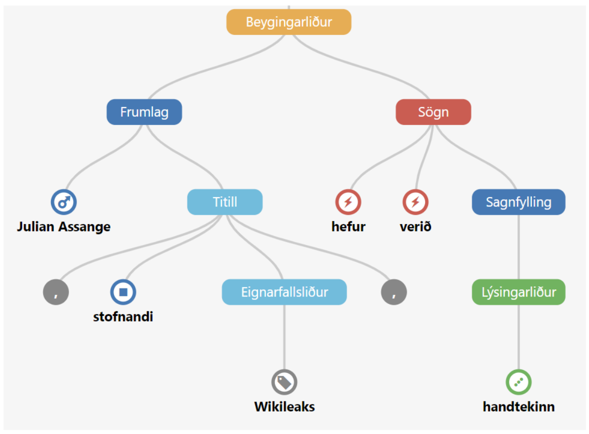

Figure 1: A part of a simplified parse tree, as displayed on the `greynir.is` website. The sentence is *Julian Assange, founder of Wikileaks, has been arrested*. (*Beygingarliður*, *Frumlag*, *Eignarfallsliður*, *Sögn*, *Sagnfylling*, *Lýsingarliður*) = (*IP*, *Subject*, *Possessive Phrase*, *VP*, *Predicative Complement*, *Adjective Phrase*).

The website is refreshed every 30 minutes with new articles from the major Icelandic news outlets. The news articles can be browsed, and parse trees for individual sentences can be viewed in a graphical format by clicking on them (see Figure 1). Sentences that the system is not able to parse are marked in red. The system shows the names and titles of people who have been mentioned in recent news articles, the definitions of named entities, and geographical locations that occur in recent news. This information is picked up from tokens and parse trees.

Apart from browsing news articles, users are able to enter their own text and have it parsed and checked by the system. As with news articles, the parse tree of a user-entered sentence can be displayed by clicking on the sentence.

9 Evaluation

On the basis of the current CFG, our parser is able to derive at least one parse tree from about 90% of sentences in our database of 8.7 million sentences extracted from news articles. The remaining 10% are divided between sentences in languages other than Icelandic, number-oriented text in currently unparsable format such as sports results and data tables, sentences containing severe spelling or grammar errors, and valid sentences which our CFG does not yet cover.

To estimate the accuracy of our CFG and the parser, we hand-annotated 80 randomly selected sentences from news articles using a simplified, syntactically bracketed annotation scheme similar to that of the Penn Treebank. The trees output from our parser were converted from their internal representation, corresponding directly to the CFG, to the simplified scheme via a set of mapping rules. Spelling and grammar errors were corrected before parsing. Results for this small test corpus, using the *evalb* tool (Sekine and Collins, 2013), are shown in Table 2.

Recall	**Precision**	**F-measure**	**Average Crossing**
71.15%	72.67%	71.90%	4.18

Table 2: *evalb* results for 78 hand-annotated sentences.

Out of 80 sentences, two could not be parsed as they contain syntactic structures not presently covered by the CFG. Out of 1,444 word tokens, 66 (4.6%) were OOV of which the de-compounding algorithm handles 35.

Many errors are caused by wrong attachment of PPs and subclauses, or when NPs from phrases or clauses deeper in the tree are erroneously attached as objects of verbs in the main clause, instead of correctly identifying a complement clause or PP as the object.

Both error types affect all intermediate levels in the tree, and thus lower *evalb*'s reported scores severely. We continue to work on our grammar and our parsing system to address these errors, as well as developing a gold standard of parsed news text for more accurate evaluation.

10 Conclusion

We have presented a system consisting of an open-source, wide-coverage CFG for Icelandic, and an accompanying parser. Our system demonstrates that it is practical to develop a wide-coverage CFG for an MRL, such as Icelandic, with a parsing system that performs well enough for real-world use cases. Such a system can be used *inter alia* for information extraction from news websites, grammar correction, and to generate dependency annotated corpora as well as treebanks for training deep neural network-based parsers.

References

Qaiser Abbas. 2016. Morphologically rich Urdu grammar parsing using Earley algorithm. *Natural Language Engineering* 22(5):775–810. https://doi.org/10.1017/S1351324915000133.

Kristín Bjarnadóttir. 2012. The Database of Modern Icelandic Inflection. In *Proceedings of the workshop "Language Technology for Normalization of Less-Resourced Languages", SaLTMiL 8 – AfLaT*. Istanbul, Turkey, LREC 2012. http://www.lexis.hi.is/kristinb/lrec2012-dmii.pdf.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation* (2):597–620. https://doi.org/10.1007/s11168-004-7431-3.

Ted Briscoe, Claire Grover, Bran Boguraav, and John Carroll. 1987. A Formalism and Environment for the Development of a Large Grammar of English. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 2*. Milan, Italy, IJCAI'87. http://dl.acm.org/citation.cfm?id=1625995.1626017.

Aoife Cahill. 2008. Treebank-Based Probabilistic Phrase Structure Parsing. *Language and Linguistics Compass* 2(1):18–40. https://doi.org/https://doi.org/10.1111/j.1749-818X.2007.00046.x.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM* 13(2):94–102. https://doi.org/10.1145/362007.362035.

Robert Gaizauskas, Mark Hepple, Horacio Saggion, Mark A. Greenwood, and Kevin Humphreys. 2005. SUPPLE: A Practical Parser for Natural Language Engineering Applications. In *Proceedings of the Ninth International Workshop on Parsing Technology*. Vancouver, Canada. http://aclweb.org/anthology/W05-1527.

Donald Hindle. 1989. Acquiring Disambiguation Rules from Text. In *Proceedings of the 27th Annual Meeting on Association for Computational Linguistics*. Vancouver, Canada, ACL '89. https://doi.org/10.3115/981623.981638.

Anton Karl Ingason, Hrafn Loftsson, Eiríkur Rögnvaldsson, Einar Freyr Sigurðsson, and Joel C. Wallenberg. 2014. Rapid Deployment of Phrase Structure Parsing for Related Languages: A Case Study of Insular Scandinavian. In *Proceedings of the* 9th *International Conference on Language Resources and Evaluation*, Reykjavik, Iceland, LREC 2014. http://www.lrec-conf.org/proceedings/lrec2014/pdf/855_Paper.pdf.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall, 2nd edition.

Alexandra Kinyon and Carlos A. Prolo. 2002. A Classification of Grammar Development Strategies. In *Proceedings of the 2002 Workshop on Grammar Engineering and Evaluation - Volume 15*. COLING-GEE '02. https://doi.org/10.3115/1118783.1118790.

Peter Ljunglöf and Mats Wirén. 2010. Syntacic Parsing. In N. Indurkhya and F. J. Damerau, editor, *Handbook of Natural Language Processing*, CRC Press. 2nd edition.

Hrafn Loftsson and Eiríkur Rögnvaldsson. 2007. IceParser: An Incremental Finite-State Parser for Icelandic. In *Proceedings of the* 16th *Nordic Conference of Computational Linguistics*. Tartu, Estonia, NODALIDA 2007. https://aclweb.org/anthology/papers/W/W07/W07-2419/.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. http://dl.acm.org/citation.cfm?id=972470.972475.

Eiríkur Rögnvaldsson, Anton Karl Ingason, Einar Freyr Sigurðsson, and Joel Wallenberg. 2012. The Icelandic Parsed Historical Corpus (IcePaHC). In *Proceedings of the* 8th *International Conference on Language Resources and Evaluation*, Istanbul, Turkey, LREC 2012. http://www.lrec-conf.org/proceedings/lrec2012/pdf/440_Paper.pdf.

Elizabeth Scott. 2008. SPPF-Style Parsing From Earley Recognisers. *Electronic Notes in Theoretical Computer Science* 203:53 – 67. https://doi.org/10.1016/j.entcs.2008.03.044.

Elizabeth Scott and Adrian Johnstone. 2010. Recognition is Not Parsing - SPPF-style Parsing from Cubic Recognisers. *Science of Computer Programming* 75(1-2):55–70. https://doi.org/10.1016/j.scico.2009.07.001.

Satoshi. Sekine and Michael J. Collins. 2013. The evalb software. `http://cs.nyu.edu/cs/projects/proteus/evalb`.

Masaru Tomita. 1986. *Efficient parsing for natural language*. Kluwer Academic Publishers, Boston.

Reut Tsarfaty, Djamé Seddah, Sandra Kübler, and Joakim Nivre. 2013. Parsing Morphologically Rich Languages: Introduction to the Special Issue. *Computational Linguistics* 39(1):15–22. https://doi.org/10.1162/COLI_a_00133.

Fei Xia. 2001. *Automatic grammar generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania. https://dl.acm.org/citation.cfm?id=934569.

D. H. Younger. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control* 10(2):189–208. https://doi.org/10.1016/S0019-9958(67)80007-X.