Human Language Technologies – The Baltic Perspective
A. Utka et al. (Eds.)

doi:10.3233/978-1-61499-442-8-184

Syntactic N-gram Collection from a Large-Scale Corpus of Internet Finnish

Jenna KANERVA [a,1], Juhani LUOTOLAHTI [a],
Veronika LAIPPALA [b] and Filip GINTER [a]
[a] *Department of Information Technology, University of Turku, Finland*
[b] *Department of French Studies, University of Turku, Finland*

Abstract. In this paper, we report on the development of a large-scale Finnish Internet parsebank, currently consisting of 1.5 billion tokens in 116 million sentences. The data is fully morphologically and syntactically analyzed and it has been used to extract flat and syntactic n-gram collections, as well as verb-argument and noun-argument n-grams. Additionally, distributional vector space representations of the words are induced using the *word2vec* method. All n-gram collections as well as the vector space models are made available under an open license.

Keywords. Finnish, syntactic parsing, n-grams, syntactic n-grams, large-scale

Introduction

Large-scale tagged text corpora and n-gram collections have been the traditional workhorse of corpus linguistics as well as the source of data for many natural language processing applications. The Internet has become an inexhaustible source of text material covering an inclusive number of languages and topics. The high coverage of the Internet makes it an appealing source for a corpus development, and consequently large-scale projects, such as WaCKy [1], have been established to build web corpora. With the recent progress in syntactic parsing algorithms, both in terms of speed and accuracy, it has become feasible to enrich the large corpora with a full syntactic analysis, building *parsebanks*.

In this paper, we report on the development of a large-scale Finnish Internet parsebank, currently consisting of 1.5 billion tokens in 116 million sentences. The text of the corpus is gathered from the plain text webpage data made available by the CommonCrawl[2] Internet crawl project using language recognition to detect Finnish text. The text is further de-duplicated on the document level and filtered to contain a "clean" narrative with sentence structure. Every sentence in the collection is then fully morphologically and syntactically analyzed using the parsing pipeline of Haverinen et al. [2].

The syntactic analysis follows the Stanford Dependencies (SD) scheme [3] with minor modifications to accommodate Finnish-specific phenomena. In addition to the ba-

[1]Corresponding author: Jenna Kanerva, University of Turku, Joukahaisenkatu 3-5, 20014 Turun yliopisto, Finland, e-mail: jmnybl@utu.fi
[2]http://commoncrawl.org

sic parse trees, the parsebank also contains the additional dependencies defined in the *non-basic (enhanced)* variants of the SD scheme. These encode non-tree relations — most importantly conjunct propagation and external subjects — resulting in dependency graphs rather than trees (see Figure 1). The enhanced analyses are produced using the machine learning method of Nyblom et al. [4].

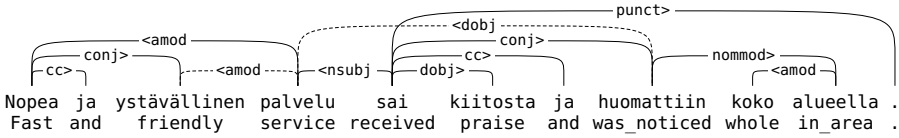

Figure 1. Conjunct propagation as defined in the non-basic variants of the SD scheme. The additional dependencies are marked with dashed lines. The example can be translated as *The fast and friendly service received praise and was noticed in the whole area*.

1. Data Preparation

1.1. Data Source

The typical starting point for gathering Internet data would be to launch a web crawl using a tool dedicated to gathering text-rich pages, such as SpiderLing[3], and restrict the crawl to the national domain `.fi`. Crawling Internet data is a time-consuming task with many complexities which are not immediately apparent, and the restriction to the national top-level domain naturally prevents a number of relevant domains from being reached. As an alternative approach, we therefore use the CommonCrawl[4] dataset containing 6.6 billion web pages. The crawl data is available as an Amazon Public dataset and can be accessed as Hadoop sequence files from the Amazon Elastic Compute Cloud and the Amazon Elastic MapReduce cluster. The crawl data is available as both plain text and HTML. The CommonCrawl data is a general-purpose Internet crawl and only a tiny fraction of it is of interest to us: Finnish pages with a sufficient proportion of text suitable for parsing.

The plain text is stored in the Hadoop sequence files as key–value pairs, where the key is an URL and the value is the content downloaded from the URL as a raw text. For every available plain text file, we iterated through each key–value pair and detected its language based on the first 400 bytes using the Chromium compact language detector package[5], preserving only Finnish pages for further processing. The choice of the language detection component was primarily motivated by its speed, since the language detection was by far the most time consuming step when gathering the initial data from CommonCrawl.

The plain text data contains catalogs, lists, menus, and the like, mixed together with the actual text of interest, and is thus not suitable for parsing as is. To gather clean text from the data, it is filtered line-wise to discard the lines of text which are not a part of any sentence. For each line we calculated the following features: The number of tokens, the

[3] http://nlp.fi.muni.cz/trac/spiderling

[4] http://commoncrawl.org, the 2012 version was available when the processing was initiated

[5] https://code.google.com/p/chromium-compact-language-detector

number of tokens recognized as Finnish by the OMorFi Finnish morphological analyzer [5,6], the number of special character tokens (non-alphanumeric characters), the number of numerical tokens, whether the line begins with an uppercase letter and whether it ends with a sentence-terminal character, such as a comma or an exclamation mark.

On a small test sample of the raw text, we manually established the parameters such that for a line to be preserved, it must be over 5 tokens long, have over 60% of its tokens recognized as Finnish, cannot contain more than 20% of numerical tokens and more than 30% of special character tokens.

The plain text lines do not necessarily follow the sentence boundaries and a single sentence can be spread across multiple lines as a part of a larger text block possibly consisting of many sentences. Thus the lines within a single block (i.e. all lines until an empty or removed line is reached) are either accepted as separate text blocks or alternatively concatenated into bigger text blocks if found relevant due to sentence boundaries. If the line starts with an uppercase word recognized as Finnish by OMorFi and ends in a sentence-terminal character, it was determined to consist of complete sentences and no concatenation was needed. Otherwise, the line was considered to be a potential part of a bigger text block and concatenated with the surrounding lines to form a complete text block with continuous sentences. After concatenation, the tokens still not forming complete sentences were stripped from the start and the end of the new block. All clean text blocks extracted from a single document were then saved together with their URL to preserve the source of a particular part of the corpus and to maintain the document structure.

Internet data contains a significant proportion of duplicate documents. To remove these duplicates, a hash was created for each unique sentence in the data and documents found to contain more than 90% of previously seen sentences were discarded.

The final text corpus after post-processing and de-duplication contains 1.5 billion tokens in 116 million sentences from approximately 4 million URLs. These sentences were subsequently parsed using the parsing pipeline described in the following section.

1.2. Dependency Parsing Pipeline

The clean text is parsed using the dependency parsing pipeline[6] of Haverinen et al. [2]. This pipeline consists of a statistical sentence splitter and tokenizer from the *OpenNLP*[7] toolset, followed by the *OMorFi*[8] morphological analyzer [5,6], the *HunPOS*[9] tagger [7], and finally the *mate-tools*[10] graph-based dependency parser [8]. All of these tools have been trained on the Turku Dependency Treebank [2]. The performance of the pipeline, measured on the TDT test set (with a gold standard sentence splitting and tokenization) is shown in Table 1. The best labeled attachment score (LAS) reported in the literature for the TDT test set is 83% using the beam-searched transition-based parser of Bohnet et al. [9], which also carries out the POS and morphological tagging jointly with parsing. The improvement of roughly 2pp in terms of LAS is however offset by a marked, nearly 10-fold decrease in the parsing speed, especially in the early experimental version of

[6] http://turkunlp.github.io/Finnish-dep-parser
[7] https://opennlp.apache.org/
[8] https://code.google.com/p/omorfi
[9] https://code.google.com/p/hunpos
[10] https://code.google.com/p/mate-tools

the parser available at the time the parsing run was executed, and we therefore used the faster, if somewhat less accurate parser.

Table 1. The performance of the parsing pipeline as measured on the TDT test set (Table adapted from [2]).

Metric	Measured	Accuracy [%]
Labeled attachment score (LAS)	Governor + Dependency type	81.01
Unlabeled attachment score (UAS)	Governor	84.97
Dependency type accuracy	Dependency type	89.53
Lemmatization accuracy	Lemma	91.8
Main part-of-speech tagging	POS	94.4
Fine-grained tagging	All morphological tags	89.8
Full morphology	Lemma + all morphological tags	87.3

1.3. Predicting Additional Dependencies

On top of the parse trees, the machine learning-based method of Nyblom et al. [4] is used to enrich the syntactic analysis by including the additional dependencies defined in the non-basic variants of the SD scheme [10]. These enhanced analyses include the propagation of coordinated elements, the explicit representation of external subjects in verbal complement structures and encoding the secondary syntactic functions of relativizers. Example of these extra dependencies are shown in Figures 1 and 2. The additional dependencies are predicted using the base parse tree as the source of features and the overall prediction performance is 93.1% in terms of F-score.

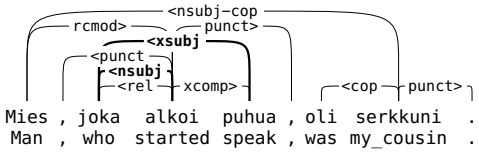

Figure 2. The secondary syntactic function of relativizer and external subject, as defined in the non-basic variants of the SD scheme. The additional dependencies are marked with dashed lines. The example can be translated as *The man who started to speak was my cousin*.

2. Flat and Syntactic N-grams

The corpus is made available in the form of *flat* and *syntactic n-grams*, in the same format in which Google recently published their collection of English flat and syntactic n-gram data derived from the Google Books corpus [11,12]. For the flat n-grams, we make available all 5-grams, i.e. fragments of five tokens together with their lemmas, part-of-speech and morphological tags. Each n-gram is associated with its count in the underlying corpus.

For syntactic n-grams, the context is defined in terms of dependency paths rather than the standard linear context, meaning that the syntactic n-grams collect together words which are close to each other in the syntactic representation, but not necessarily

in the underlying text. Following the same format as in the English syntactic n-gram collection [12], each syntactic n-gram is defined to be a single rooted connected subtree extracted from the full parse tree. Possible subtree configurations comprising from one (*arc*) to four (*quadarc*) dependencies are shown in Figure 3, when arcs, biarcs and triarcs include all possible configurations, but quadarcs are limited to a single structure, where the n-gram root has two dependents, either of which have one dependent as well. In addition to these, the n-gram collection also defines *nodes*, which are single words associated with the incoming dependency type, but not a head token.

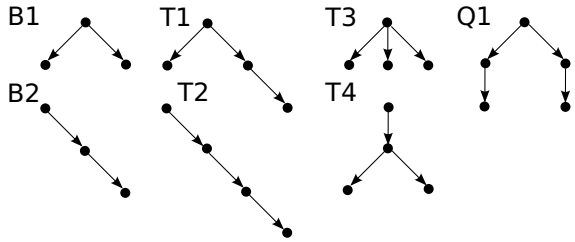

Figure 3. All possible configurations of biarcs (B) and triarcs (T) and the only quadarc structure (Q) included, without the extra functional dependents.

Instead of using the basic parse tree as the source structure, we use the richer syntactic representation defined in the non-basic variants of the SD scheme resulting in parse graphs rather than trees (as explained in Section 1.3). These additional dependencies are treated in the same manner as the base dependencies when possible (resulting in one of the configurations shown in Figure 3), but since the underlying structure is a dependency graph, all extracted n-gram structures may not be trees. Thus, the possible n-gram configurations are expanded to also include non-tree structures, where a token can have two incoming dependencies either from the same governor or from two different governors. Examples of possible non-tree configurations containing two (biarc) and three (triarc) dependencies are shown in Figure 4. Also these additional n-gram configurations are limited to contain only single rooted subgraphs, meaning that each subgraph must have only one root token. Since quadarcs are originally limited to include only one specific structure (shown in Figure 3Q), these non-tree configurations are not included in quadarcs.

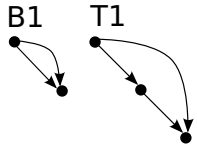

Figure 4. Example configurations of non-tree biarcs (B) and triarcs (T), without the extra functional dependents.

Words are divided into content words and functional markers to distinguish meaning bearing words from structural elements (e.g. adpositions and conjunctions). When constructing the syntactic n-grams, only the dependencies covering the content words are included and thus each tree structured n-gram of order n includes exactly $n + 1$ content words, but can include any number of additional functional markers as extra dependents.

However, in the non-tree n-gram structures one content word must have two incoming dependencies, thus always removing one content word from the n-gram configuration.

Functional markers encoding coordinating conjunctions and prepositions (dependency types *cc* and *adpos*) are always included in the n-grams, when also the particular coordination or prepositional phrase is present, whereas all other functional markers (*det*, *poss*, *neg*, *aux*, *auxpass*, *mark*, *complm* and *prt*) are included only in the extended datasets. The division between content words and functional markers follows the same principles as in the English syntactic n-gram data.

For each word in a syntactic n-gram, we provide information about its word form, lemma, main part-of-speech and all morphological tags, as well as its head in the syntactic n-gram and its dependency relation to the head word. The n-grams also preserve the relative order of the words, but naturally do not guarantee that the distance of the words remains the same, since some of the linear context words may not be included. Each unique n-gram is also associated with its count in the underlying corpus, where the minimum occurrence count of 2 is used. All configurations, from nodes to quadarcs, are made available.

Finally, we also make available verb-argument and noun-argument n-grams, whereby we gather from the corpus every configuration of a verbal or nominal predicate and all its syntactic dependents. In these datasets, all direct syntactic dependents of a predicate (also including additional dependencies) are included regardless of whether the word is seen as a content word or a functional marker. All words tagged as a verb or a noun during the parsing are considered to be predicates.

The total number of unique n-grams is given in Table 2 with varying frequency cut-off values. Only n-grams occurring at least two times in the underlying corpus are included in the current n-gram collections.

Table 2. The size of the flat and syntactic n-gram data collections in terms of unique n-grams with varying n-gram frequency cut-off values.

cut-off	5-grams	nodes	arcs	biarcs	triarcs	quadarcs	verb-args	noun-args
2	264M	25M	187M	558M	1372M	300M	48M	57M
3	53M	14M	67M	142M	305M	59M	10M	16M
5	10M	7.9M	27M	39M	68M	11M	2.2M	5.4M
7	4.2M	5.8M	16M	20M	30M	4.4M	1.0M	3.0M
9	2.4M	4.7M	12M	13M	19M	2.5M	0.6M	2.0M
11	1.7M	4.0M	8.9M	9.5M	13M	1.7M	0.5M	1.6M

3. Vector Space Embeddings of Words

Distributional vector space representations of the lexicon, such as the Latent Semantic Analysis and Random Indexing, are a useful tool in many natural language processing tasks, as they provide the means to establish in an unsupervised fashion the semantic similarity between words as well as higher units of text. To induce these representations, a large corpus of text is necessary to provide sufficient statistics on word distribution. Recently, *word2vec*, a new method to induce such vector space representations, was introduced by Mikolov et al. [13]. The method stems from the research on neural network language modeling, training a simplified neural network to predict nearby context words

given a focus word. The vector space representations are then extracted from the internal weights of the trained network. In addition to being very efficient, learning on billions of tokens worth of data in a matter of hours, the quality of the representations induced by *word2vec* has been shown to surpass a number of popular methods.

Complementary to the basic *word2vec* method, where the linear contexts of several words before and after the focus word are used when inducing the representation, we have also modified the method and its implementation to consider the syntactic context, in much the same vein as Levy and Goldberg [14]. In this case, the context is formed by the governor and the dependents of the target word, together with their mutual dependency relations. As Levy and Goldberg show, this syntactically-informed method generates qualitatively substantially different representations, capturing less topical and more functional similarity [14].

We applied both the basic and the syntactically-informed version of *word2vec* to the parsed text corpus, and make available the resulting representations for all tokens with frequency of at least 3.

4. Conclusions

The Finnish Internet Parsebank is the first large-scale fully syntactically analyzed corpus of Finnish. The present public data release consists of both flat and syntactic n-gram collections, as well as verb-argument and noun-argument n-grams, following the standard format of such collections for interoperability. The SD scheme's second layer is used when generating the n-grams. Together with the data, we also make available the open-sourced tools to generate similar n-grams from a syntactically parsed text.[11]

As the parsebank constitutes a non-trivial amount of fully morpho-syntactically analyzed recent Internet text, we expect it to be applicable in lexicographic research, morphological analysis, semi-supervised dependency parsing, distributional semantics, language modeling, and other research areas which traditionally benefit from large quantities of text analyzed to various degrees. To this end, we have applied vector space models induced from the data using the *word2vec* method to Finnish semantic role labeling [15]. In addition to the n-gram collections, we make available also *word2vec* trained vector space models induced from the parsebank. Both linear and syntactic context-based models are released.

Future work includes the development of a web-based interface for the search through the full parsebank data, as well as a possible release of the full parsed documents, pending the resolution of some outstanding legal questions.

5. Acknowledgments

This work was supported by the Kone and Emil Aaltonen Foundations. Computational resources were provided by CSC – IT Center for Science, Espoo, Finland.

[11] https://github.com/jmnybl/syntactic-ngram-builder

References

[1] Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226, 2009.

[2] Katri Haverinen, Jenna Nyblom, Timo Viljanen, Veronika Laippala, Samuel Kohonen, Anna Missilä, Stina Ojala, Tapio Salakoski, and Filip Ginter. Building the essential resources for Finnish: the Turku Dependency Treebank. *Language Resources and Evaluation*, pages 1–39, 2013.

[3] Marie-Catherine de Marneffe, Miriam Connor, Natalia Silveira, Samuel R. Bowman, Timothy Dozat, and Christopher D. Manning. More constructions, more genres: Extending Stanford Dependencies. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 187–196, Prague, Czech Republic, August 2013. Charles University in Prague, Matfyzpress, Prague, Czech Republic.

[4] Jenna Nyblom, Samuel Kohonen, Katri Haverinen, Tapio Salakoski, and Filip Ginter. Predicting conjunct propagation and other extended Stanford Dependencies. In *Proceedings of the International Conference on Dependency Linguistics (Depling 2013)*, pages 252–261, 2013.

[5] Tommi Pirinen. Suomen kielen äärellistilainen automaattinen morfologinen jäsennin avoimen lähdekoodin resurssein. Master's thesis, University of Helsinki, 2008.

[6] Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. HFST tools for morphology — an efficient open-source package for construction of morphological analyzers. In *State of the Art in Computational Morphology*, volume 41 of *Communications in Computer and Information Science*, pages 28–47. 2009.

[7] Péter Halácsy, András Kornai, and Csaba Oravecz. HunPos – an open source trigram tagger. In *Proceedings of ACL'07, Companion Volume*, pages 209–212, 2007.

[8] Bernd Bohnet. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING'10*, pages 89–97, 2010.

[9] Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428, 2013.

[10] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies manual. Technical report, Stanford University, September 2008.

[11] Yuri Lin, Jean-Baptiste Michel, Erez Aiden Lieberman, Jon Orwant, Will Brockman, and Slav Petrov. Syntactic annotations for the Google Books ngram corpus. In *Proceedings of the ACL 2012 System Demonstrations*, pages 169–174. Association for Computational Linguistics, 2012.

[12] Yoav Goldberg and Jon Orwant. A dataset of Syntactic-Ngrams over time from a very large corpus of English books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 241–247. Association for Computational Linguistics, 2013.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Workshop Proceedings of International Conference on Learning Representations*, 2013.

[14] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, 2014. Association for Computational Linguistics.

[15] Jenna Kanerva and Filip Ginter. Post-hoc manipulations of vector space models with application to semantic role labeling. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 1–10, 2014.