CHAPTER 21

Beyond Counting Syntactic Hits

Erwin R. Komen
Radboud University Nijmegen, SIL-International, E.Komen@ru.nl

ABSTRACT

Linguists who would like to make use of the increasing number of syntactically annotated text corpora in their research can use existing tools to find and count instances of the syntactic constructions they are interested in. Software supporting linguists in their work should also make it possible to build databases of search results where each hit is accompanied by a number of calculated (or manually addable) features. The stand-alone CorpusStudio program is able to provide this help, since it allows queries and feature calculations to be defined in the XQuery language. The web application of CorpusStudio, which is still under development, aims to have comparable functionality but with an easier accessibility. The main aim of this chapter is to demonstrate why software should go beyond counting syntactic hits.

Keywords: syntax, corpus research, XQuery

21.1 Introduction

A linguist who is interested in studying a particular syntactic construction in a language can do so by manually or programmatically looking through a number of texts in that language. It is the availability of syntactically annotated texts that makes this latter programmatic approach possible.

There are quite a number of programs and even web applications linguists can use to find instances of the syntactic construction they are interested in.[1] Studies conducted by linguists, however, involve more than locating constructions that satisfy particular conditions. Two other important aspects of a study are: (a) keeping a number of related searches together in a search

[1] Some of these programs are mentioned later on in this article.

How to cite this book chapter:
Komen, E. R. 2017. Beyond Counting Syntactic Hits. In: Odijk, J and van Hessen, A. (eds.) *CLARIN in the Low Countries*, Pp. 259–268. London: Ubiquity Press. DOI: https://doi.org/10.5334/bbi.21. License: CC-BY 4.0

'project' that can be stored and retrieved to improve replicability, and (b) annotating search results automatically or semi-automatically with information that can be gleaned from the search hits. While the latter activity is an integral part of corpus linguists' everyday research, little support in terms of software is available.

This chapter discusses and exemplifies the kind of facilities beyond those for counting syntactic hits that linguists would greatly appreciate in syntactic corpus research programs. The observations discussed are based on experience with the CorpusStudio and Cesax programs, which have so far been used in historical linguistics, second language acquisition and information structure research for Indo-European (Dutch, English, Welsh) as well as Caucasian (Chechen, Lak, Lezgi) languages (Komen 2014; Komen et al. 2014; Los and Dreschler 2012; van Vuuren 2013). The CorpusStudio application allows researchers to formulate and execute syntactic searches, store them in a 'Corpus Research Project', and annotate the search results with features that are determined programmatically.

21.2 The Linguist

I would like to underscore the idea that linguists want to do more than finding syntactic constructions by considering what kinds of questions linguists ask when studying the syntax of a particular language. Linguistics is a broad research area, but I would like to focus on the research on syntax and information structure where annotated corpora are used. The important questions that researchers in this area ask are summarised in (1):

(1) a. Under what circumstances does construction 'x' occur, and, coupled with this question, what are the distinguishing properties of this construction?[2]

b. How does the occurrence of construction 'x' depend on genre, dialect or author, and how did the construction develop over time?

Finding instances of construction 'x' and counting them in a particular corpus is a good first step towards answering these questions, but more should and could be done. Let me illustrate this with a real-life research question. Consider the examples from the 'standard' conditional construction in Dutch in (2a) and the alternative conditional inversion in (2b).

(2) a. Nou **als** je niet kijkt op een paar miljoen
well if you not look on a few million
dan kun je dus stellen dat de eerste drie kernactiviteiten
then could you therefore posit that the first three nuclear.activities
nagenoeg evenveel budget ter beschikking hebben.
almost equal budget to disposal have
'If a few million aren't too important, then one could say that the first three activities have more or less the same budget.' **[fn000056:0047]**

b. **Heeft** u de partners gevonden **dan** begint het eigenlijk pas
have you the partners found then starts it actually only
want dan moet er een projectvoorstel geschreven worden.
because then must there a project.proposal written become

[2] I use the term 'construction' here to denote a constellation of syntactic units. Any construction in this sense can be defined by making use of hierarchy and linear order of units that are identified by syntactic labels, possibly together with limitations on the content of these units.

'It is only when the partners have been found, that the matter actually starts. That's when the project proposal needs to be written.' **[fn000056:0147]**

Suppose that linguists want to investigate the occurrence of the conditional inversion as opposed to the standard if-then conditional: they would at least want to know the numbers, so that they can figure out whether one of the two constructions is more or less exceptional. The numbers can be found by searching through syntactically annotated texts. Tools that facilitate syntactic searches are, for instance, the web applications PaQu[3] (Parse and Query; see chapter 23) and GrETEL[4] (see chapter 22) as well as the Windows version of CorpusStudio.[5] All three search engines handle the corpus of Dutch texts in which the examples above occur: the Corpus of spoken Dutch (Oostdijk et al., 2002).[6]

It should be obvious from the examples in (2) that the two conditional syntactic constructions differ.[7] If linguists want to find all relevant results, they would probably need to write two different queries. Even if the researchers' focus is not on the 'standard' conditional, they would want to have the number of their occurrences for the sake of comparison. The two different queries do, however, belong to the same 'research project', which is why it would be of great help for linguist-users of the search software to have these queries stored together – and they could do with some metadata too, identifying what the goal of the queries is, for instance. One possibility to reach this goal would be to keep searches and documentation in different files, but store them in a single project directory. This is a good approach, but keeping all relevant information together in one structured file (e.g. in XML format) makes it even more transparent, prevents potential errors and promotes clarity.

In line with the general research question in (1), linguists would like to know under which circumstances the conditional inversion occurs. They want to know whether its occurrence depends on linguistic factors, as in (1a), extra-linguistic factors, as in (1b), or a combination of the two. Table 21.1 identifies a number of linguistic and extra-linguistic features that linguists would probably want to have for each hit.

How do linguists investigating the conditional inversion enrich their list of hits with the information they need? They could look through all the hits identified by a syntactic search program, and then find all the relevant information for each of the hits manually, by checking the texts. But such an approach is error-prone, and, if there are no other reasons to check the texts manually, should be avoided. As for a programmatic approach, a few authors have suggested that XQuery could be used to extract the information that is required (Bouma, 2008; Bouma and Kloosterman, 2007; Yao and Bouma, 2010). The XQuery language is well suited to this task, since it allows the user to work with variables and functions (Boag et al., 2010). The language would, however, form an obstacle for linguists who are less familiar with computer languages. Applications that support XQuery, then, should consider supporting easier query definition methods for some, while allowing the use of XQuery's fuller capabilities for others.

Suppose, now, that the features mentioned in Table 21.1 have been determined for each of the hits. This gives the linguists basic data to do their research. They would be much helped if it were possible to divide the results into groups, the categories of which depend on the features that are

---

3 http://portal.clarin.nl/node/4182

4 http://nederbooms.ccl.kuleuven.be/eng/

5 http://erwinkomen.ruhosting.nl/software/CorpusStudio

6 Other systems that facilitate syntactic search queries are outside the scope of this chapter. Among them is the CLARIN-developed PML Tree Query web application, which aims for dependency treebanks and makes use of the PML query language (Mírovský et al., 2010)

7 No numerical results of the searches discussed here are given, since the focus of this chapter is not on this particular syntactic phenomenon, but on the question of how best to help a linguist wanting to research this and similar phenomena.

Type	Feature	Value
Linguistic	`TenseType`	Is this a periphrastic (‘*heeft ... gevonden*’) or a simple tense?
	`FirstSize`	The size of the first part of the condition (the protasis).
	`Pre`	The kind of element (if any) preceding the conditional (e.g. the *nou* ‘well’ in (2a)).
	`ParaPosition`	The position within the paragraph (start, middle, end).
	`FirstStatus`	The information status of the first part of the condition: does it link back to the preceding context or is it new?
Extra-linguistic	`AuthorName`	Who is the author (perhaps the use of the conditional inversion is linked to a limited number of authors?)?
	`AuthorAge`	Would the conditional inversion be an innovation (young authors) or a remnant from the past (old authors)?
	`AuthorDialect`	Is the conditional inversion linked to particular dialects?
	`TextType`	Is it linked to a particular type of text?
	`TextDate`	The publication date of the text.

Table 21.1: Features that could be relevant for choosing a conditional inversion.

calculated. That would give them a fast way to check the hypotheses that underlie the determination of the features in the first place.

It would also be nice if they could divide their search into two parts. In a first step, they could first look for instances of the conditional inversion and the standard conditional, enrich them with the features listed in Table 21.1, and store them in some kind of database. They could manually check and adapt features such as ‘`ParaPosition`’ and ‘`FirstStatus`’, since these may not be determinable automatically with enough accuracy. They would need to have access to the hits in their context at this point.

The next step would be to formulate and test hypotheses that determine the choice between a standard conditional and a conditional inversion. This step would require to take the data in the result database from the previous step as input. It would be quite natural to implement this step by using the same machinery as in the previous step.

Once all of this has been done, the linguists have quite likely reached a point where they want to make use of programs such as R or SPSS to test statistical models of their hypotheses. The corpus research software should allow the data to be exported in such a way that it can be used by statistics programs.

The facilities that corpus research software should provide to help linguists address the kind of research questions in (1) are summed up in (3).

(3) a. Find and count instances of syntactic constructions.
b. Provide figures that allow for the calculation of relative frequencies: the number of words, clauses, and texts that have been searched.[8]
c. Store the search results separately, so that features can be added to them.
d. Calculate required features automatically as much as possible.

[8] Calculation of relative frequencies is left to the linguist, since the point of reference by which absolute frequencies should be divided may be taken differently depending on the purpose.

e. Allow for researchers to adjust or add features manually.
f. Allow results to be divided into categories that are data-dependent.
g. Allow results to be divided into groups that are metadata-dependent.
h. Allow using a collection of (annotated) results as input for one or more other queries.
i. Have the queries and the feature calculations that belong to one research project together in one place, allowing interchange and replicability.
j. Allow users to enrich texts with features.
k. Allow exporting the data for use in statistics programs and for publications.

Facility (3a) looks for and finds instances of the construction, and (3b) adds information to allow for a good quantitative study. The facilities in (3c–e) allow researchers to equip each 'hit' with as many features as are needed to help answer the research question. Facilities (3f–g) help provide more insight into how the results are divided in terms of aspects of the data itself or the metadata. Facility (3i) promotes the exchange of research projects and contributes to replicability. Facility (3c) allows for the process in (3a–h) to be divided into two parts: one where a database with hit-feature combinations is created, and one where the results in this database are divided into adjustable groups. Facility (3k) provides the connection with a possible next step: a statistical analysis.

21.3 Current Software

Software that addresses points (3a,g) partly or completely has been made or continues to be made. The programs produced or enhanced for CLARIN-NL and CLARIN Flanders are no exception. A consortium of organisations and universities developed the Corpus Hedendaags Nederlands tool and later the OpenSONAR tool (Oostdijk et al., 2002; Reynaert et al., 2014).[9] The Nederlab web application provides access to a huge (and growing) amount of Dutch texts (Brugman et al., 2016).[10] All interfaces address (3a,g), some address (3f) partly, but none of these currently feature syntactic searches.

21.3.1 Web-based CLARIN Tools for Syntactic Research

Two tools that have been supported by CLARIN that do allow for some kind of syntactic search are PaQu and GrETEL. PaQu has been developed by the University of Groningen.[11] It not only incorporates online access to the Alpino parser of Dutch, but also provides search interfaces that allow the user to define queries in XPath. Satisfying facility (3a), search results can be downloaded and are accompanied by some metadata.

The GrETEL tool allows searches in a number of different Dutch corpora as well as in Afrikaans corpora.[12] Its user interface vastly differs from that of PaQu: searches are formed on the basis of a real-life example provided by the linguist. This means that researchers do not need to have in-depth knowledge of what goes on inside the search engine. Augustinus et al. (2012) explain that their search engine uses XPath for the actual searches. The XPath code produced by GrETEL can, in fact, be used without changes in the PaQu web interface. The GrETEL application addresses point (3a), it allows the downloading of all the hits, and it has an option that provides a table with the counts divided per treebank; this table partly addresses facilities (3b,g).

[9] `opensonar.clarin.inl.nl` and `opensonar-cgn.science.ru.nl`
[10] `http://www.nederlab.nl`
[11] `http://www.let.rug.nl/alfa/paqu`
[12] `http://nederbooms.ccl.kuleuven.be/eng/gretel`

21.3.2 Windows-based CorpusStudio and Cesax

Two Windows-based programs combine into a set of tools that address most of the ambitious goals defined in (3): CorpusStudio and Cesax (Komen et al., 2013). Figure 21.1 shows how the programs cooperate.

The CorpusStudio program works with Corpus Research Projects (CRPs), XML definitions of queries, and metadata that together describe a research project. It allows the defining of searches in XQuery, which means that users can define variables and functions and use these in their queries. CorpusStudio works on XML text corpora that are located on the user's computer, addressing (3a) fully. The search results it provides contain the total number of words and sentences of the texts being searched, allowing for (3b), the calculation of relative frequencies. Dividing the results in a data-dependent way – (3f) – is possible through a CorpusStudio-specific built-in XQuery function. Division of the results on the basis of metadata is only possible to a limited extent, so point (3g) is addressed only partly. The results can be turned into a separate XML database, and each 'hit' can be accompanied by user-definable features, addressing (3c). The extensive capabilities of the XQuery language, and the fact that it allows for user-defined functions in particular, facilitate calculation of such hit-dependent features in a comprehensive but relatively user-friendly way, as per (3d). The Cesax program allows for working with the kinds of result databases produced by CorpusStudio, so that the features can be adapted manually as per (3e). Points (3h) and (3j) are also taken care of by CorpusStudio and Cesax respectively. And where GrETEL offers an example-based definition of queries, Cesax and CorpusStudio contain a 'query wizard' that allows users to base a query on key elements of an example sentence in the corpus. Keeping queries, feature calculations, and metadata together in one research project, as indicated by (3i), is addressed fully by CorpusStudio (this was actually one of the main reasons to write the program in the first place).

The stand-alone version of CorpusStudio does, unfortunately, come with a number of shortcomings. It is platform-dependent, since it only works on Windows. Its speed depends very much on the characteristics of the computer on which it is running, but it is not very fast. And while CorpusStudio could be adapted to work with XML texts in the FoLiA format, this is not facilitated

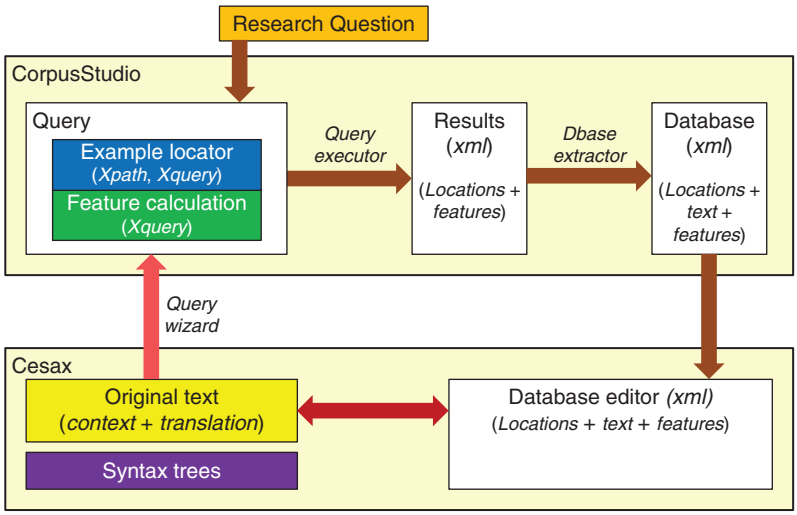

Figure 21.1: Cooperation between CorpusStudio and Cesax.

directly.[13] A disadvantage related to its nature as a stand-alone program is the fact that a copy of the corpus to be researched needs to be held on each user's own machine.[14] Where text corpora are being adapted, one may quickly lose track of where the most up-to-date version is located. Most of these disadvantages are alleviated in the web-version of CorpusStudio.

21.4 The Web Application

The stand-alone CorpusStudio Windows program has partly been re-written as a web application.[15] The key components of the web application are shown in Figure 21.2.[16]

The core of the application is the 'Query Executor', a Java application that accepts a Corpus Research Project and executes the XQuery code from that project on a corpus of XML texts (in the FoLiA or the TEI-Psdx format).[17] The CrpxProcessor divides the query execution workload over the available processors; the more processors, the faster the query execution. The CrpxProcessor can be run as a stand-alone application, but it is used as part of a web service within the CorpusStudio web application: the `/crpp` search service.

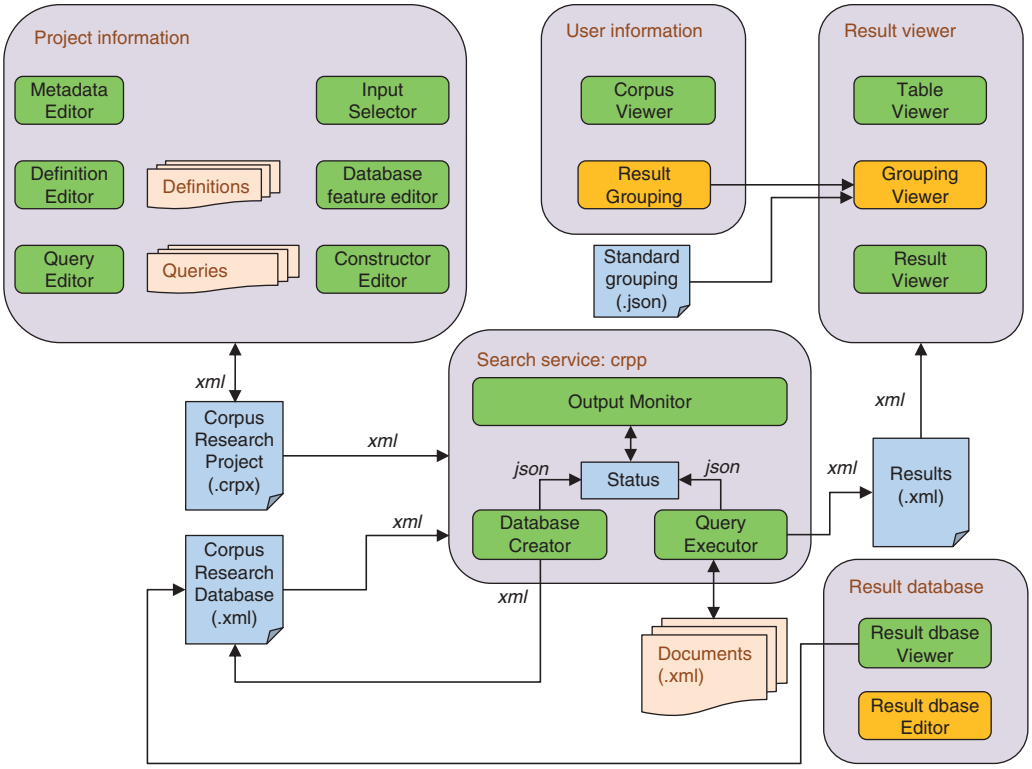

Figure 21.2: Principal components of the CorpusStudio web application.

[13] Texts in the FoLiA format can be converted to the TEI-Psdx format in Cesax and then processed.

[14] This is a particular shortcoming of CorpusStudio, not of stand-alone programmes as such. A reviewer pointed out that the Dact programme, for instance, is a stand-alone cross-platform application that supports working on remote corpora with remote parsing servers (van Noord et al., 2013). See `http://rug-compling.github.io/dact/`

[15] `http://www.clarin.nl/node/2095`

[16] The source code of the application is available at `https://github.com/ErwinKomen`.

[17] A number of other formats can be converted into FoLiA or Psdx through the Cesax programme.

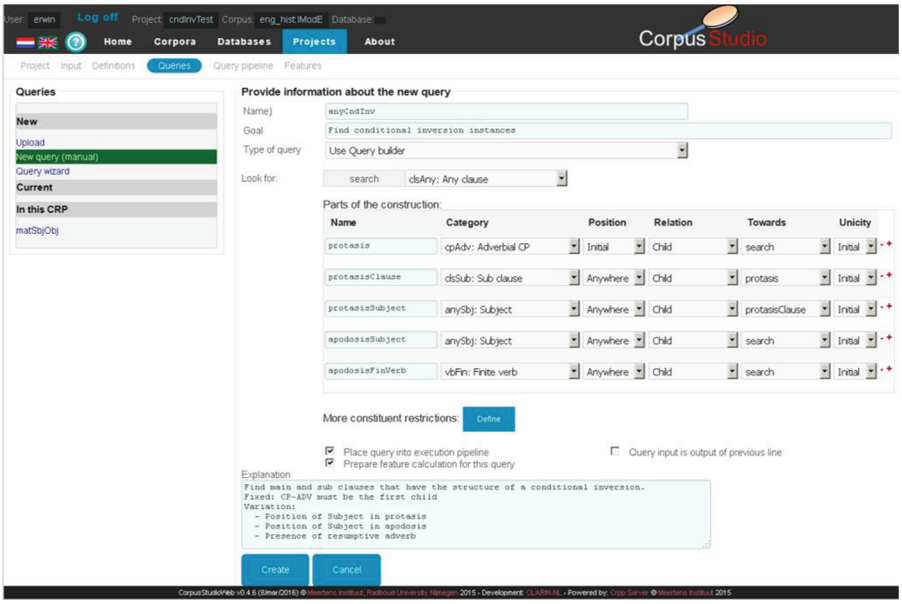

Figure 21.3: Query input wizard.

The user of the CorpusStudio web application works with the `/crpstudio` service; this provides the interface between the user on the one hand, and the server on the other hand. The server contains the syntactically annotated XML text corpora that can be searched, the user's Corpus Research Projects (`.crpx` files), the user's search results and possibly the user's result databases. The `/crpstudio` service allows for the definition of the information stored in the Corpus Research Projects: the metadata of the project (Metadata Editor); the XQuery variables, definitions and queries (Definition and Query Editor); the hierarchy between the queries (Constructor Editor); and the features that need to be calculated if the output is a database (Database feature editor).

The 'corpora' part of the `/crpstudio` service lists the corpora that are available in the web application (the Corpus Viewer) and allows defining metadata-dependent result groupings. The 'dbases' part of the service makes interaction with the result databases possible. Once a research project has been executed, its results are available in the result viewer, which also allows downloading them.

The version of the CorpusStudio web application that has been delivered at the end of 2015 still suffers a number of limitations compared to its stand-alone Windows counterpart; there are, for instance, limitations on metadata-dependent grouping of results and on working with result databases The implementation of a query wizard has started in 2016 and consists of two phases: (1) a query input wizard that allows easy input of queries, which are subsequently translated into XQuery, and (2) a system level that forms a shell around XQuery, allowing users to define and adapt queries without the need for them to know any XQuery (queries are translated into XQuery only just before execution).

The query input wizard is currently being implemented, and Figure 21.3 gives an idea of its intermediate state. The main idea is that the user can: (1) name and identify constituents and their relations towards one another, (2) stipulate additional relations between the named constituents, and (3) formulate feature definitions on top of the standard ones (the latter of which are the labels of each of the identified constituents, and the text of these constituents). More information on the current status of the program, including the second phase (of easy access to XQuery), will be made available online.[18]

[18] See the 'About' section of the web application: `http://www.clarin.nl/node/2095`.

Most importantly, the program has extended the CLARIN infrastructure with a syntactic research tool that allows interested linguists to make use of points (3a–i) in their research[19]

21.5 Discussion and Conclusions

Current tools available to linguists who are interested in doing syntactic research on annotated corpora allow finding and counting syntactic constructions. This chapter takes the conditional inversion as an example, and shows that more software help can be given to address the kinds of questions a linguist asks. This chapter argues that a researcher would want to annotate all the instances of constructions like the standard conditionals and the conditional inversion with features, taking the research beyond counting syntactic hits.

Users of the stand-alone CorpusStudio have already shown that the availability of this kind of sofware influences the research process itself: instead of focusing on finding one particular syntactic construction, the creation of feature databases that can again serve as the input to the search process leads to initially broader searches that make use of quite specific feature calculation functions.

Software that facilitates the intended process could make use of the query language XQuery, since it not only allows searching through syntactically annotated corpora, but also allows calculating the values of the features a linguist may be interested in. The existing CorpusStudio stand-alone Windows program makes use of this query language but has the drawbacks of most stand-alone applications. It is platform-dependent and does not easily help other linguists to work with the same corpus. This chapter mentions the first version of the CorpusStudio web application, a web-based version of the Windows program. While it does not yet offer all the facilities a researcher would like to make use of, it brings the kind of corpus-based syntactic research advocated in this chapter a step closer to users of the CLARIN infrastructure.

Acknowledgements

I am grateful to CLARIN-NL, who shared my vision and financed the development of the web application. My Radboud colleagues Meta Links and Sanne van Vuuren have provided useful comments on CorpusStudio and shared their ideas about the web application with me. I owe a lot of thanks for fruitful discussions on technical aspects to my Meertens Institute colleagues Matthijs Brouwer, Erik Tjong Kim San, Hennie Brugman, and Jan Pieter Kunst.

References

Augustinus, Liesbeth, Vandeghinste, Vincent & van Eynde, Frank. 2012. 'Example-based treebank querying'. Paper presented at *Eighth international conference of language resources and evaluation (LREC2012)*, Istanbul, Turkey.

Boag, Scott, Chamberlin, Don, Fernández, Mary F., Florescu, Daniela, Robie, Jonathan & Siméon, Jérôme. 2010. *XQuery 1.0: An XML Query Language (Second Edition)*: W3C Recommendation, <http://www.w3.org/XML/Query/#specs>.

[19] I leave a detailed discussion of the CorpusStudio web application to another platform, since the main goal of this chapter is to show that syntactic research would be served by software that goes beyond counting hits. Much of the information one would be interested in is available in the CorpusStudio manual. The application is not a typical database application: it searches through physical texts. The search speed of the web application is proportional to the number of processor cores the server it runs on makes available, since the search of each text takes place in a separate thread. Eight cores give a speed improvement of 7.8 compared to one core. The speed of the benchmark project ('V2_test_versie11') running on the Old English YCOE corpus (1.5 million words) is 18 min, 49 sec on the Windows stand-alone application (2 GHz processor), while it takes 22.64 sec on the web application that makes use of 20 cores.

Bouma, Gosse. 2008. XML information extraction with Xquery: processing wikipedia and Alpino trees. Groningen: Information science, university of Groningen.

Bouma, Gosse & Kloosterman, Geert. 2007. Mining syntactically annotated corpora with XQuery. In *Proceedings of the Linguistic Annotation Workshop*. Prague, Czech Republic: Association for Computational Linguistics.

Brugman, Hennie, Reynaert, Martin, Sijs, Nicoline van der, Stipriaan, René van, Tjong Kim Sang, Erik, Bosch, Antal van den, Kunst, Jan Pieter, Zeeman, Rob, Kooij, Dieuwertje, Brussee, Ineke, Brouwer, Matthijs, Kemps-Snijders, Marc & Bennis, Hans. to appear. Nederlab: towards a single portal and research environment for diachronic Dutch text corpora. In *Language resources and evaluation conference (LREC 2016)*. Portorož (Slovenia).

Komen, Erwin R. 2013. Corpus databases with feature pre-calculation. In *Proceedings of the twelfth workshop on treebanks and linguistic theories (TLT12)*. Sandra Kübler, Petya Osenova & Martin Volk (eds), 85–96. Sofia, Bulgaria: The institute of information and communication technologies, Bulgarian academy of sciences, <http://www.bultreebank.org/TLT12/TLT12Proceedings.pdf>.

Komen, Erwin R. 2014. Chechen extraposition as an information ordering strategy. In *Information structure and reference tracking in complex sentences*. Rik van Gijn, Dejan Matić, Jeremy Hammond, Saskia van Putten & Ana Vilacy Galucio (eds), 99–126. Amsterdam: John Benjamins.

Komen, Erwin R., Hebing, Rosanne G. A., van Kemenade, Ans & Los, Bettelou. 2014. Quantifying information structure changes in English. In *Information Structure and Syntactic Change in Germanic and Romance Languages*. Kristine Gunn Eide & Kristin Bech (eds), 81–110. Amsterdam, New York: John Benjamins.

Los, Bettelou & Dreschler, Gea. 2012. The loss of local anchoring: From adverbial local anchors to permissive subjects. In *Rethinking Approaches to the History of English*. Terttu Nevalainen & Elizabeth Closs Traugott (eds), 859–872. New York: Oxford University Press.

Mírovský, Jiří, Mladová, Lucie & Žabokrtský, Zdeněk. 2010. 'Annotation tool for discourse in PDT'. Paper presented at *Proceedings of the 23rd International Conference on Computational Linguistics: Demonstrations*.

Oostdijk, Nelleke, Goedertier, W. , Eynde, F. van, Boves, Lou , Martens, J.-P. , Moortgat, M. & Baayen, Harald. 2002. 'Experiences from the Spoken Dutch Corpus Project'. Paper presented at *Proceedings of the 3rd international conference on language resources and evaluation (lrec2002)*, Las Palmas.

Reynaert, Martin, Camp, Matje van de & Zaanen, Menno van. 2014. 'OpenSoNaR: user-driven development of the SoNaR corpus interfaces'. Paper presented at *Proceedings of the 25th International Conference on Computational Linguistics (Coling 2014)*, Dublin, Ireland.

van Noord, Gertjan, Bouma, Gosse, Van Eynde, Frank, de Kok, Daniël, van der Linde, Jelmer, Schuurman, Ineke, Tjong Kim Sang, Erik & Vandeghinste, Vincent. 2013. Large Scale Syntactic Annotation of Written Dutch: Lassy. In *Essential Speech and Language Technology for Dutch*. P. Spyns & J. Odijk (eds), 147–164.

van Vuuren, Sanne. 2013. Information structural transfer in advanced Dutch EFL writing: a cross-linguistic longitudinal study. In *Linguistics in the Netherlands 2011 [AVT30]*. Suzanne Aalberse & Anita Auer (eds), 173–187. Amsterdam: John Benjamins.

Yao, Xuchen & Bouma, Gosse. 2010. 'Mining Discourse Treebanks with XQuery'. Paper presented at *Ninth international workshop on treebanks and linguistic theories (TLT9)*, Tartu, Estonia.