CHAPTER 20

Creating Research Environments with BlackLab

J. de Does[a], J. Niestadt[b] and K. Depuydt[c]

[a]INT, Leiden, Netherlands, jesse.dedoes@ivdnt.org, [b]INT, Leiden, Netherlands, jan.niestadt@ivdnt.org, [c]INT, Leiden, Netherlands, katrien.depuydt@ivdnt.org

ABSTRACT

The *BlackLab* search engine for linguistically annotated corpora is a recurring element in several CLARIN and other recent search and retrieval projects. Besides the core search library, we have developed the *BlackLab Server* REST web service which makes it easy for computational linguists and programmers to write anything from quick analysis scripts to full-fledged search interfaces, and the *AutoSearch* application which allows nontechnical linguistic researchers to index and search their own data.

This chapter describes the motivation for developing the BlackLab platform, how it has been used in actual research, and sketches future developments which will make it a more powerful tool for the creation of research environments.

20.1 Introduction: Why BlackLab and BlackLab Server?

There are several excellent linguistic corpus search engines that support the creation of corpus retrieval systems: the Sketch Engine (Kilgarriff et al., 2004) is a superb product, the Corpus Workbench (Evert and Hardie, 2011) is widely used to create search interfaces for corpora, cf. for instance (Borin et al., 2012; Nygaard et al., 2008), and there are more recent alternatives like Corpuscle (Meurer, 2012), and Poliqarp (Janus and Przepiórkowski, 2007).

Nevertheless, there were reasons to look for alternatives. In the context of the CLARIN and CLARIAH research infrastuctures, we need a versatile platform that supports the creation of research environments that can be either generic or tailored to specific needs. Of course, the search engine at the heart of such a platform should still be powerful, scalable, efficient and feature-rich, but other requirements are just as important: the core components should be easy to maintain and

How to cite this book chapter:

de Does, J, Niestadt, J and Depuydt, K. 2017. Creating Research Environments with BlackLab. In: Odijk, J and van Hessen, A. (eds.) *CLARIN in the Low Countries*, Pp. 245–257. London: Ubiquity Press. DOI: https://doi.org/10.5334/bbi.20.

extend because of clear APIs and modular design, and, because of the simplicity of both the library and the server API, it should be easy to develop custom front ends and extensions.

Our choice to develop a new corpus retrieval platform that uses Lucene as the underlying search engine has the advantage that we can profit not only from the active development of the Lucene core, but also from Lucene-based products like Solr and Elasticsearch to implement new features.

20.2 BlackLab, BlackLab Server and AutoSearch

20.2.1 The Design of BlackLab

We had the following objectives in mind while designing BlackLab:

1. Modularity and flexibility, enabling, for instance, easy implementation of new document formats (for instance, a FoLiA[1] indexer has been added in less than a day)
2. Strict separation of front end and back end
3. Scalability of the indexing core only bounded by the excellent Lucene scalability
4. Incremental indexing
5. Support for Corpus Query Language (CQL),[2] a widely-used linguistic query language
6. Development in a modern, mainstream language (Java) enabling fast and robust development of the engine itself and of retrieval applications that use the engine
7. Open source

Extending the Basic Lucene Indexing and Retrieval Model

Lucene is at the heart of BlackLab. Each indexed document becomes a Lucene document, and metadata fields such as title and author become Lucene fields. The document content is indexed in a more sophisticated way: token and character positions are stored. This enables highlighting of search results in the original content.

BlackLab extends this basic mechanism in several ways:

Multiple token attributes Multiple properties can be stored for each word. A common use case is to store the word form, lemma and part of speech, but any other type of information is possible. Each of these properties is stored in a separate Lucene field, and BlackLab transparently combines these fields while searching.

Querying BlackLab uses Lucene's SpanQuery classes for querying. This allows the most flexibility in matching complex patterns. The SpanQuery classes included with Lucene were not enough to support the more advanced features of Corpus Query Language, so we had to extend them. The extension of the Span query mechanism supports features like the repetition operator (e.g. for finding a sequence of two or more adjectives) and searching inside XML tags. Besides the Corpus Query Language (abbreviated as CQL or CQP), BlackLab also supports the (basic) Contextual Query Language (SRU/CQL).

Content store Retrieving and (optionally) highlighting the original content is made possible by efficiently storing the original indexed (XML) content in the 'content store'. The data is stored using gzip compression, which saves a lot of disk space.

---

[1] `http://proycon.github.io/folia/`

[2] The Corpus Workbench site has a great introduction to CQL: `http://cwb.sourceforge.net/files/CQP_Tutorial/` Note that BlackLab supports the most important features, but not yet all of the features.

Forward index For quickly displaying keyword-in-context (KWIC) views, sorting and grouping hits on context, and counting occurrences of terms in whole documents or in the vicinity of a set of hits, a specialized data structure called a forward index has been implemented. The forward index is really the complement to Lucene's reverse index: whereas Lucene answers questions of the form 'where in my documents does the word X occur?', the forward index is optimized to answer questions of the form 'what word occurs in document Y at position Z?'

20.2.2 Features of BlackLab Server

BlackLab Server was developed for two reasons: to provide a clean back end for the corpus front end, a large part of which is written in JavaScript, and to make it as easy as possible to carry out quick corpus analyses from any scripting language without compromising the speed of the BlackLab Java code. BlackLab Server is a REST web service: it responds to URL requests in either JSON or XML format. It is implemented as a Java servlet that will run, for instance, in Apache Tomcat. It provides several different search modes, such as: search for occurrences of a word, search for documents containing a word, show a snippet of text from a document, or retrieve the full original document. In addition to sorting results, it also allows you to group results by many criteria, including the context of the hits found (e.g. the word to the left of the hit). Some important aspects of its design are:

- Smart caching of search results
- The user can decide to use blocking or nonblocking request mode
- Protection against overtaxing the server. BlackLab Server tries to prevent (intentional or unintentional) server abuse by monitoring searches and terminating ones that are taking too long or consuming too many resources.

20.2.3 AutoSearch

For researchers who are not computational linguists or programmers, but would like to be able to quickly search their annotated texts, we have developed BlackLab AutoSearch. This application allows end users to simply upload text data in a supported format (today, FoLiA or TEI). It is then indexed on our servers, after which it may be searched using Corpus Query Language.

If the user does not have FoLiA or TEI data yet, but rather text data in another format (e.g. Word, PDF, HTML or ePub), we have also developed OpenConvert[3], which allows users to convert their plain text data into FoLiA or TEI, and run it through a (simple) tagger/lemmatiser for Dutch. In the future, we would like to incorporate this functionality into AutoSearch, to streamline the process as much as possible.

20.2.4 Performance

An elaborate comparison to other corpus retrieval systems is outside the scope of this chapter. Benchmarking would be easier if standard query and datasets were available for this purpose. Nevertheless, to obtain an indication of the performance level, we tagged and lemmatized the *DUTCH PARLIAMENTARY PROCEEDINGS 1814-2012* dataset, consisting of about 700 million tokens,

[3] Both AutoSearch and OpenConvert can be found in our CLARIN portal at `https://portal.clarin.inl.nl/`. OpenConvert is also available on GitHub: `https://github.com/INL/OpenConvert`. AutoSearch should soon be available under `https://github.com/INL/` as well; send us a message if you are interested.

query	hits	CWB	BlackLab
[pos="AA.*"] [lemma="krokodil"]	73	13s	**48ms**
"beslissing" "om" "niet" "te" [pos="VRB.*"]	8	**60ms**	273ms
[pos="NOU.*"] "om" "te" [pos="VRB.*"]	76660	50s	**22s**
[pos="VRB.*"]{7}	1672	38s	**17s**
[pos="AA.*"]+ [pos="NOU.*"] [pos="VRB.*=fin.*" & lemma='doen'] [pos="AA.*"]+ [pos="NOU.*"]	95	**24s**	25s

Table 20.1: Query times on Parliamentary Proceedings Corpus.

available from the Dutch Data Archiving and Networked Services (DANS)[4], and indexed the data with BlackLab and with CWB[5].

The performance on some example queries is illustrated in Table 20.1. We found the systems to be roughly comparable in performance, with some queries running faster in BlackLack (command line query tool) and others in CWB (command line tool `cqp`).

20.3 Using BlackLab and BlackLab Server to build your own research environment

This hands-on section explains how to use BlackLab and BlackLab server to build simple applications.

20.3.1 Indexing Data with BlackLab

BlackLab can index any textual data, but we have focused on using it with XML. Several XML formats (including popular corpus formats FoLiA and TEI[6]) are supported out-of-the-box, and it is easy to create custom versions of indexers or add support for a new XML format.

XML corpus formats generally have an XML tag for each word, with tags or attributes for the different properties of the word (such as lemma and part of speech). BlackLab indexes each property in its own Lucene field, and automatically combines these fields while searching, so you can construct complex queries that specify constraints on different properties as needed.

20.3.2 Using BlackLab Directly from Java

Before introducing the web service, we start with the BlackLab Java API, so that we can compare the two. Here is some example code that uses the BlackLab API directly to search for a Corpus Query Language query:

```
// Convert word array to string
String words(List<String> words) {
    return StringUtil.join(words, " ");
}

final static File PATH_TO_MY_CORPUS = new File("/tmp/bla/");

// Search and show hits
public void search(String cqlQuery) {
```

[4] https://easy.dans.knaw.nl/ui/datasets/id/easy-dataset:51640

[5] BlackLab 1.3.1, cwb 3.0.0, 12-core Intel(R) Xeon(R) CPU E5-2630 0 @ 2.30GHz, 128G ram.

[6] http://www.tei-c.org/

```
    try (Searcher searcher = Searcher.open(PATH_TO_MY_CORPUS)) {
        TextPattern tp = CorpusQueryLanguageParser.parse(cqlQuery);
        Hits hits = searcher.find(tp);
        for (Hit hit: hits) {
            Kwic kwic = hits.getKwic(hit);
            Document document = searcher.document(hit.doc);
            String title = document.get("title");
            System.out.println(words(kwic.getLeft("word")) + " ["
                    + words(kwic.getMatch("word")) + "] "
                    + words(kwic.getRight("word")) + " (" + title + ")");
        }
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

As we shall see below, using BlackLab Server results in very similar code.[7]

20.3.3 BlackLab Server

As stated, BlackLab Server allows you to use BlackLab from any programming language, and we will give two examples of this here.

20.3.3.1 A Simple Example

Here is a simple Python example of searching a corpus for a CQL pattern (`[pos="a.*"] "fox"`), i.e. the word 'fox' preceded by an adjective and displaying a simple textual KWIC view with document titles.

```
import urllib
import json

def words(context):
    """ Convert word array to string. """
    return " ".join(context['word'])

def search(cqlQuery):
    """ Search and show hits. """
    url = "http://example.com/blacklab/mycorpus/hits?patt="
            + urllib.quote_plus(cqlQuery)
    with (urllib.open(url)) as f:
        response = json.loads(f.read())
    hits = response['hits']
    docs = response['doc-infos']
    for hit in hits:
        # Show the document title and hit information
        doc = docs[hit['doc-pid']]
        print words(hit['left']) + " [" + words(hit['match']) + "] " +
                words(hit['right']) + " (" + doc['title'] + ")"

# "Main program"
search('[pos="a.*"] "fox"')
```

We have translated this basic example into other languages as well (including JavaScript, R, PHP, Perl, C# and Ruby). These may be found online[8].

[7] The complete Java BlackLab API documentation can be found at `http://inl.github.io/BlackLab/apidocs/`.

[8] `http://github.com/INL/BlackLab-server/wiki/Using-BlackLab-Server-from-different-languages`

20.3.3.2 Slightly More Complex BlackLab Server Example

This example draws bar charts of the collocations of certain words in some author's works.

To start, here is a simple HTML page: just a search form and a div to render our chart to. It includes jQuery (for convenience), Google Charts (for drawing the chart) and our own JavaScript file, `blacklab-server.js`.

```
<html>
  <head>
    <script type="text/javascript" src="jquery.js"></script>
    <script type="text/javascript" src="https://www.google.com/jsapi"></script>
    <script type="text/javascript" src="blacklab-server.js"></script>
  </head>
  <body>
    <h1>Zen and the Art of Collocations</h1>
    <form onsubmit="search(); return false;">
      Show words that occur within 5 words of <input id='word' type='text' />
      <input type='submit' value='Update' />
    </form>
    <div id="chart" style="width: 900px; height: 500px;"></div>
  </body>
</html>
```

The next example, `blacklab-server.js`, sends a request to the server, counts context words in the response, and draws the chart. To be precise, when the form is submitted, the `search` function is called, which builds a CQL query, glues it to a URL, and retrieves that URL. When the server responds, the `handleResults` function iterates over the hits in the response object, counting words in the left and right contexts, and renders the resulting word frequency data using Google Charts.

```
google.load("visualization", "1", {packages:["corechart"]});

var whichContext = "word"; // which collocations? e.g. word/lemma/pos/..

function search() {
    var cqlQuery = '"' + jQuery("#word").val() + '"';
    var url = "http://example.com/blacklab/mycorpus/hits?filter=author:pirsig&"
            + "context=5&patt=" + encodeURIComponent(cqlQuery);
    jQuery.get(url, handleResponse);  // AJAX call to BlackLab Server
}

function handleResponse(response) {
    // Count context words for each hit
    var wordFreq = {};
    jQuery.each(response['hits'], function (index, hit) {
        countWords(hit['left'], wordFreq);  // left context
        countWords(hit['right'], wordFreq); // right context
    });

    // Draw Google Chart
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Word');
    data.addColumn('number', 'Frequency');
    jQuery.each(wordFreq, function (word, freq) {
        data.addRows([[word, freq]]);
    });
    data.sort([{column: 1, desc: true}]);
    var chart = new google.visualization.BarChart(jQuery('#chart').get(0));
    chart.draw(data);
}

function countWords(context, wordFreq) {
    jQuery.each(context[whichContext], function (index, word) {
```

```
        if (!wordFreq[word]) wordFreq[word] = 0;
        wordFreq[word]++;
    });
}
```

20.4 Using BlackLab and BlackLab Server for Linguistic Research

We summarize how BlackLab has been used for research, and analyze the requirements that can be deduced from these experiences. Finally, a use case based on the *Letters as Loot* corpus illustrates how a (small) research environment created with BlackLab server can support historical linguistic research.

20.4.1 Projects Using BlackLab

IMPACT The IMPACT[9] project was about enhancing the accessibility of historical documents in library collections. To demonstrate the potential of using linguistic resources for this purpose, INL developed a Lucene-based search engine, intended to exploit linguistic data in full text retrieval of library collections.

CLARIN search and develop An SRU endpoint implementation for BlackLab was developed to integrate the search engine in the CLARIN-NL research infrastructure.

Corpus Gysseling The Corpus Gysseling[10] contains almost all known 13th-century Dutch text. It is the principal source for the Dictionary of Early Middle Dutch.

Corpus Hedendaags Nederlands (Corpus of contemporary Dutch) The Corpus Hedendaags Nederlands (CHN) is a first step towards a monitor corpus for contemporary Dutch, intergrating corpora gathered by INL in the 1990s with more recent material. The corpus is available to the research community as part of the CLARIN-NL research infrastucture[11].

OpenSoNaR OpenSoNaR is an online system that allows for analyzing and searching the large scale Dutch reference corpus SoNaR. SoNaR is a 500-million-word reference corpus of contemporary written Dutch for use in different types of linguistic (including lexicographic) and language technology research and the development of applications. In this CLARIN-NL project, a powerful corpus exploration user interface was developed for the SONAR-500 corpus, using BlackLab server as a back end[12].

Letters as Loot The *Letters as Loot* corpus is a corpus of 1,033 Dutch letters from the 17th and 18th century. They were sent home from abroad by sailors and others, but also abroad by those staying behind who needed to keep in touch with their loved ones. Many letters did not reach their destinations: they were taken as loot by privateers and confiscated by the High Court of Admiralty during the wars fought between the Netherlands and England.

This corpus, to which metadata from the research programme's database were added, was lemmatised, PoS-tagged and provided with elaborate search facilities by the Institute for Dutch Lexicology[13].

Early Modern English corpora at Northwestern University Phil Burns of Northwestern University has created an experimental corpus search site[14] that is powered by BlackLab. At

[9] http://www.impact-project.eu/, http://www.digitisation.eu

[10] http://gysseling.corpus.taalbanknederlands.inl.nl/

[11] http://corpushedendaagsnederlands.inl.nl/

[12] http://opensonar.clarin.inl.nl

[13] http://brievenalsbuit.inl.nl

[14] http://devadorner.northwestern.edu/corpussearch/

present the corpus of Shakespeare's plays, the TCP ECCO corpus (Eighteenth Century Collections Online), the TCP Evans corpus (Evans Early American Imprints), and the Shakespeare His Contemporaries corpus (Early Modern English Drama) are publicly searchable. Martin Mueller (Professor of English & Classics) has written about his experiences with the application (Mueller, 2013).

20.4.2 Research and education based on BlackLab corpora

The following research uses the BlackLab query engine:

- OpenSoNaR and CHN have been used in teaching corpus linguistics in courses at Leiden university and Utrecht university
- Marc van Oostendorp and Nicoline van der Sijs had a very interesting presentation[15] on the history of *na* vs. *naar* at the LUCL workshop *Effects of Prescriptivism in Language History, 21-22 January 2016*[16], using (among others) the *Letters as Loot* corpus
- Den Ouden (2014) looks for transitive verbs in intransitive contexts
- Kiers (2014) investigated periphrastic versus synthetic comparatives in Dutch and Polak (2015) made an analysis of the influence of phonetical context on the distributions of the suffixes -ig, -erig, -achtig, respectively, a Master's Thesis and a Bachelor's thesis relying on data obtained from the Corpus Hedendaags Nederlands.

We also mention some research that makes use of the corpora mentioned in another way, sometimes simply because the research was performed before the corpus was online. We list this type of research because of the requirements it poses:

- The *Letters as Loot* corpus has been used as the main source of information for a groundbreaking study in historical sociolinguistics (Rutten and Van der Wal, 2014). Most analyses are based on careful manual work, which remains indispensable in many cases. In many cases the analysis requires comparing frequencies of different phonological and grammatical phenomena.
- Nobel (2013) investigates diminutives in the *Letters as Loot* corpus.

20.4.2.1 Requirements emerging from these experiences

Teaching sessions Elaborate corpus retrieval sessions with the OpenSoNaR user interface at Utrecht University in courses given by Jan Odijk yielded, among others, the following requirements:

Querying 1. Define variables, or at least equality restrictions that can for instance query for word repetitions[17]; 2. Improve part-of-speech querying, so regular expression matching is not needed to select a part-of-speech feature[18]; and 3. Enable parametrized queries from input list

Grouping and sorting 1. Grouping and filtering by arbitrary combinations of metadata, and arbitrary functions of hit text, e.g case-insensitive grouping of word forms; 2. Relative frequencies of groups with respect to subcorpus size; and 3. Custom sorting criteria.

User data and annotation 1. Persistent query history per user; 2. Metadata upload (in CMDI format); and 3. Support for categorization of results, subsequently usable for grouping, sorting, etc.

---

[15] https://prezi.com/ofiy5m-a6vbe/na-and-naar/?utm_campaign=share&utm_medium=copy

[16] http://nederl.blogspot.nl/2015/11/21-22-january-2016-effects-of.html?m=1

[17] This refers to a CQL feature not yet implemented by BlackLab.

[18] This can be solved easily by a different indexing scheme.

Export of query results 1. Tab-separated export format: separate all fields by tabs; options for simple and extended part-of-speech export; options to export metadata; and 2. Export of CMDI metadata describing the result export: including query, filters, grouping criteria, number of documents/hits/groups in results. This should be uploadable to the application to reproduce the result.

Research experiences Out of the above-mentioned research, the following requirements can be deduced:

- Many would have benefited from flexible options to export data in the user interface.
- In several cases, some elementary statistics incorporated in the user interface could have been helpful in the course of investigation, although the complete investigation requires types of analysis that cannot be foreseen in a generic interface. In (Kiers, 2014), a simple option to analyze the distribution over time (in the style of Google n-grams) would have helped the researcher; in (Polak, 2015), analyses are more complex, but direct data export to R from BlackLab Server[19] would have helped
- Relative frequencies instead of absolute counts in grouping results
- Grouping by arbitrary combination of metadata attributes, and custom criteria defined by user
- Cleaning result data, adding information to it both on a document level and on a token-by-token basis (this is in agreement with the desideratum of result categorization by users, mentioned above) would benefit many researchers. Nobel (2013), for instance, discards results from letters where spelling does not give her enough information to deduce the phonological realisation of the diminutive suffix in a reliable way
- Comparison of number of results from two (or more) queries, distributed over metadata properties, would also have benefited Van Oostendorp and Van der Sijs
- An option to involve lexical data often seems called for, enabling options like 'give me intransitive occurrences of verbs that normally require a direct object'. This corresponds roughly to the parametrized queries mentioned before.

In most of these cases, we can argue that the use of BlackLab Server could make it very easy to implement the requested features. For some features, extensions to BlackLab server are necessary, but mostly of a rather simple nature, e.g. an option to return all relevant subcorpus sizes corresponding to metadata grouping criteria.

20.4.3 Use case: signs and sounds in the Letters as Loot *corpus*

For this case study, we have developed a small research environment to start exploring how we can support the kind of research that has been conducted in (Rutten and Van der Wal, 2014), most of it before the corpus appeared online. To this end, we compare some results from chapter 2 ('Sounds and signs - From local to supralocal usage') of (Rutten and Van der Wal, 2014) to results obtained from querying the corpus and analyzing the query results.

It is obvious that automatic retrieval from corpora cannot replace careful manual analysis in many cases. For instance, the analysis of the orthographical representation of etymologically distinct long *e*'s[20] requires information which is simply not present in the annotated corpus.

---

[19] `https://github.com/INL/BlackLab-server/wiki/Using-BlackLab-Server-from-different-languages#r`

[20] Cf. section 2.4.5 in (Rutten and Van der Wal, 2014): 'Many Dutch dialects, the southern ones in particular, maintain the phonological difference between lengthened *ē* out of originally short vowels in open syllables, and *ê* out of the West Germanic diphtong **ai*'.

20.4.3.1 H-Dropping in the 17th Century: First Case Study

Many dialects from the south and the south-west of the Dutch language area are characterised by the absence of the phoneme *h*, as in, *and* instead of *hand*. In the texts, this may result in deletion or prothesis of *h*.

As we have seen, contrasting two result sets is a desideratum emerging from corpus research. As a test case, we have used the BlackLab Server API and Google Charts (in a similar vein to the simple concordance example (section 20.3.3.1) to implement this functionality in a simple way (cf. Figure 20.1). Our environment will consist of a search and grouping form, a bar chart, and a simple concordance view.

In the example, we contrast the number of hits of the corpus query

```
[lemma != "h.*"] [lemma="h.*" & word = "[aeo].*"]
```

(indicating *h*-dropping[21]) to the query specifying orthographic expression of *h*:

```
[lemma != "h.*"] [lemma="h.*" & word = "h.*"]
```

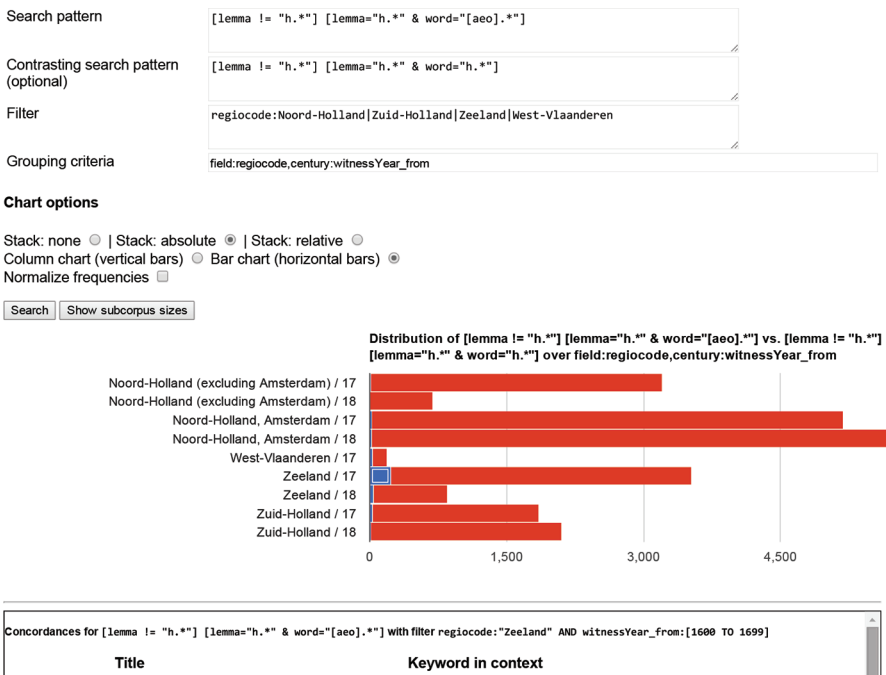

Figure 20.1: H-dropping 1: absolute frequencies.

[21] The restriction on the previous word is there to avoid situations like 'hier om', where both word parts have the lemma 'hierom'.

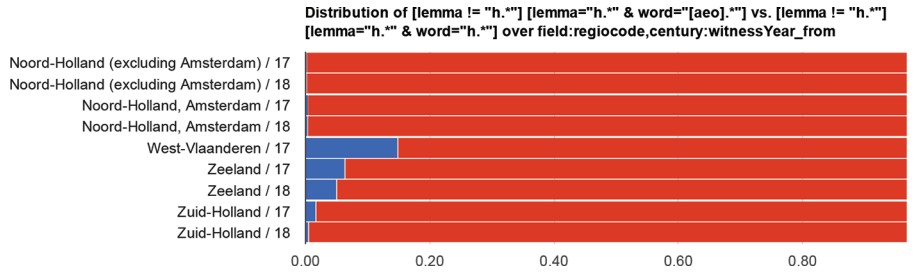

Figure 20.2: H-dropping 2: relative frequencies.

The result (cf. also Figure 20.2) indicates clearly that *h*-dropping is a southern (Zeeland and Western Flanders) phenomenon, and is more predominant in the 17th than in the 18th century. Comparing to the manual results (133 cases of *h*-dropping in the 17th-century Zeeland corpus), we should note that counts are not identical because we are using a larger corpus and the selection criteria are different, but the observed tendency is in agreement with the Rutten-Van der Wal results.

Summarizing, we are able to reproduce this type of analysis comparatively easily. The fact that our query results are, with respect to the phenomenon we are looking for, neither complete nor quite clean, does not impair their usefulness as a quick way to analyze a tendency. For more thorough analysis, one would need the result categorization feature discussed above.

20.4.3.2 Loss of Final -e

One of the most salient changes in the history of Dutch is apocope of final schwa, a linguistic phenomenon that also occurred in English and to a lesser extent in German. By the 17th century, many dialects and particularly Holland dialects had a high proportion of schwa-less forms.

The change shows prominently in first person singular forms of verbs. In nouns, forms with final *-e* are hard to distinguish from plurals with loss of final *n*.

Hence:

```
[lemma="ik" & word=".*[ck].*"]
[pos="VRB" & word=".*e" & word != ".*[td]e"
           & lemma != "doen|gaan|staan|slaan|zien|zijn"]
[pos != "VRB"]
```

is a way to find word forms with final *e*, and

```
[lemma="ik" & word=".*[ck].*"]
[pos="VRB" & word!=".*e"
           & lemma != "doen|gaan|staan|slaan|zien|zijn"]
[pos != "VRB"]
```

finds their *e*-less counterparts, cf. Figure 20.3.

20.5 Conclusions and Future Plans

We are still improving BlackLab and its related projects: scaling BlackLab up to ever larger corpora, making sure even complex searches remain fast, and adding useful features. We are interested in looking at distributed search and multi-corpus search, both for speeding things up and keeping larger datasets manageable. We are considering to integrate BlackLab with Solr or Elasticsearch to enable this functionality. Another feature on our wishlist is the ability to search tree- and graph-like

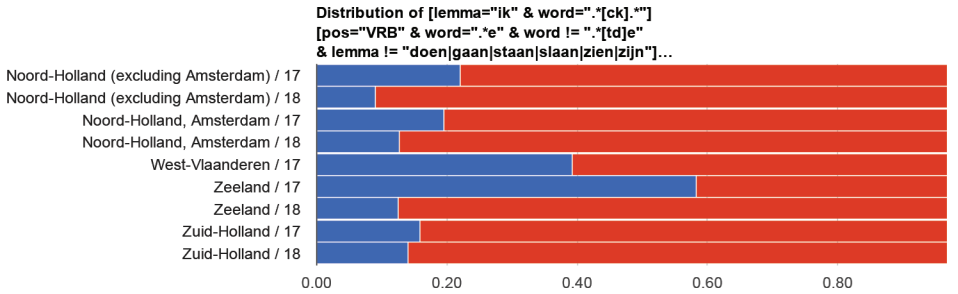

Figure 20.3: Dropping of final *e* in first person singular.

structures (e.g. treebanks). We will look at both of these desirable features as part of CLARIAH. Other CLARIAH objectives that fit in very well with the requirements that emerge from the research discussed in this chapter are so-called 'Chaining Search' (serial combination of searches in heterogeneous datasets, e.g. a corpus and a lexicon) and adding comprehensive support for dealing with subcorpora, included those defined by document metadata uploaded by researchers.

In the near future, we would like to create a library for talking to BlackLab Server from one or more popular programming languages, which could abstract away the last few technical details, making things even easier. Support for statistical explorations and visualizations should be enhanced, cf. for instance (Speelman, 2014). As has been discussed before, the aim is not to develop a monolithic application that satisfies all requirements, but rather the development of a platform that supports quick development of the analysis scripts and user interface elements that are necessary for a research use case.

References

Borin, L., Forsberg, M. and Roxendal, J. (2012). Korp – the corpus infrastructure of språkbanken, *Proceedings of LREC 2012. Istanbul: ELRA*, pp. 474–478.

Den Ouden, M. (2014). *Theta rollen en argument drop*, B.S. Thesis, Universiteit van Amsterdam, the Netherlands.

Evert, S. and Hardie, A. (2011). Twenty-first century corpus workbench: Updating a query architecture for the new millennium, *Proceedings of the Corpus Linguistics 2011 Conference*, Birmingham, UK.

Janus, D. and Przepiórkowski, A. (2007). Poliqarp: an open source corpus indexer and search engine with syntactic extensions, *Proceedings of the 45th Annual Meeting of the ACL*, Stroudsburg, PA, USA, pp. 85–88.

Kiers, F. (2014). *Frequenter of meer frequent - Een corpusonderzoek naar de invloed van het Engels op de trappen van vergelijking in het Nederlands*, Master's thesis, Universiteit Utrecht, the Netherlands.

Kilgarriff, A., Rychly, P., Smrz, P. and Tugwell, D. (2004). The sketch engine, *Proc EURALEX 2004*, Lorient, France, pp. 105–116.

Meurer, P. (2012). Corpuscle - a new corpus management platform for annotated corpora, *in* G. Andersen (ed.), *Exploring Newspaper Language: Using the web to create and investigate a large corpus of modern Norwegian*, Benjamins.

Mueller, M. (2013). Blacklab: searching a tcp corpus by linguistic and structural criteria. **URL:** *https://scalablereading.northwestern.edu/?p=296*

Nobel, J. (2013). Small but tough – Diminutive suffixes in seventeenth-century Dutch private letters, *Taal en tongval* **65**.

Nygaard, L., Priestley, J., Nøklestad, A. and Johannessen, J. B. (2008). Glossa: a multilingual, multimodal, configurable user interface., *LREC*, European Language Resources Association.

Polak, W. (2015). *In welke fonologische context komt afleiding met de achtervoegsels -ig, -erig en -achtig voor?*, B.S. Thesis, Universiteit van Amsterdam, the Netherlands.

Rutten, G. and Van der Wal, M. (2014). *Letters as Loot. A sociolinguistic approach to seventeenth- and eighteenth-century Dutch*, John Benjamins, Amsterdam & Philadelphia.
URL: *http://www.jbe-platform.com/content/books/9789027269577*

Speelman, D. (2014). Logistic regression: A confirmatory technique for comparisons in corpus linguistics, *Corpus Methods for Semantics: Quantitative studies in polysemy and synonymy* , Vol. 43 of *Human Cognitive Processing*, John Benjamins, pp. 487–533.