NEALT

The North European Association for Language Technology

Proceedings of the Workshop on

Innovative Corpus Query and Visualization Tools

NODALIDA 2015

May 11-13, 2015
Institute of the Lithuanian Language
Vilnius, Lithuania

Cover photo '*Vilnius castle tower by night'* by Mantas Volungevičius
http://www.flickr.com/photos/112693323@N04/13596235485/
Licensed under Creative Commons Attribution 2.0 Generic
See http://creativecommons.org/licenses/by/2.0/ for full terms

Cover design Nils Blomqvist

Proceedings of the Workshop on Innovative Corpus Query and Visualization Tools at NODALIDA 2015

Editors
Gintarė Grigonytė, Simon Clematide,
Andrius Utka and Martin Volk

May 11-13, 2015
Vilnius, Lithuania

Published by

Linköping University Electronic Press, Sweden
Linköping Electronic Conference Proceedings #111
ISSN: 1650-3686
eISSN: 1650-3740
NEALT Proceedings Series 25
ISBN: 978-91-7519-035-8

Copyright

Linköping University Electronic Press Linköping, Sweden, 2015
Linköping Electronic Conference Proceedings, No. 111
ISSN: 1650-3686
eISSN: 1650-3740
URL: http://www.ep.liu.se/ecp_home/index.en.aspx?issue=111
NEALT Proceedings Series, Vol. 25
ISBN: 978-91-7519-035-8

Preface

Recent years have seen an increased interest in and availability of many different kinds of corpora. These range from small, but carefully annotated treebanks to large parallel corpora and very large monolingual corpora for big data research.

It remains a challenge to offer flexible and powerful query tools for multilayer annotations of small corpora. When dealing with large corpora, query tools also need to scale in terms of processing speed and reporting through statistical information and visualization options. This becomes evident, for example, when dealing with very large corpora (such as complete Wikipedia corpora) or multi-parallel corpora (such as Europarl or JRC Acquis).

The QueryVis workshop has gathered researchers who develop and evaluate new corpus query and visualization tools for linguistics, language technology and related disciplines. The papers focus on the design of query languages, and on various new visualization options for monolingual and parallel corpora, both for written and spoken language.

We hope that QueryVis will stimulate discussions and trigger new ideas for the workshop participants and any reader of the proceedings. The preparation of the workshop and the reviewing of the submissions has already been an inspiring experience.

All papers were peer-reviewed by three program committee members. We would like to thank all reviewers and contributors for their work and for sharing their thoughts and experiences with us.

Let us all join our forces to make corpus exploration a rewarding, entertaining, and exciting experience which will grant us ever new insights into language and thought.

May 4, 2015
Zürich

Gintarė Grigonytė
Simon Clematide
Andrius Utka
Martin Volk

Program Committee

Janne Bondi Johannessen	University of Oslo
Noah Bubenhofer	University of Zurich
Simon Clematide	University of Zurich
Johannes Graën	University of Zurich
Gintarė Grigonytė	Stockholm University
Miloš Jakubíček	Lexical Computing Ltd.
Andrius Utka	Vytautas Magnus University
Martin Volk	University of Zurich
Robert Östling	Stockholm University

Table of Contents

KoralQuery – a General Corpus Query Protocol

Joachim Bingel, Nils Diewald
Institut für Deutsche Sprache
Mannheim, Germany
`bingel,diewald@ids-mannheim.de`

Abstract

The task-oriented and format-driven development of corpus query systems has led to the creation of numerous corpus query languages (QLs) that vary strongly in expressiveness and syntax. This is a severe impediment for the interoperability of corpus analysis systems, which lack a common protocol. In this paper, we present KoralQuery, a JSON-LD based general corpus query protocol, aiming to be independent of particular QLs, tasks and corpus formats. In addition to describing the system of types and operations that KoralQuery is built on, we exemplify the representation of corpus queries in the serialized format and illustrate use cases in the KorAP project.

1 Introduction

In the past, several corpus query systems have been developed with the purpose of exploring and providing access to text corpora, often under the assumption of specific linguistic questions that the annotated corpora have been expected to help answer. This task-oriented and format-driven development has led to the creation of several distinct corpus query languages (QLs), including those mentioned in Section 3. Such QLs vary strongly in expressiveness and usability (Frick et al., 2012).

This brings several unpleasant consequences both for researchers and developers. For instance, the researcher who uses a particular system must formulate her queries in no other QL than the one used for this system, which might require additional training prior to the actual research. It might even be the case that certain research questions cannot be answered due to limitations of the QL, while the actual query system and the underlying corpus data could in fact provide results. For developers, the lack of a common protocol prevents interoperability between different query systems, for instance to forward user requests from one system to another, which may have access to additional resources.

In this paper, we present KoralQuery, a general protocol for the representation of requests to corpus query systems independent of a particular query language. KoralQuery provides an extensible system of different linguistic and metalinguistic types and operations, which can be combined to represent queries of great complexity. Several query languages can thus be mapped to a common representation, which lets users of query systems formulate queries in any of the QLs for which such a mapping is implemented (cf. Section 4). Further benefits of KoralQuery include the dynamic definition of virtual corpora and the possibility to simultaneously access several, concurrent layers of annotation on the same primary textual data.

2 Related Work

In former publications, KoralQuery was introduced as a unified serialization format for CQLF[1] (Bański et al., 2014), a companion effort focussing on the identification and theoretical description of corpus query concepts and features.

Another approach to a common query language that is independent of tasks and formats is CQL (*Contextual Query Language*) (OASIS Standard, 2013), with its XML serialization format XCQL.[2] KoralQuery differs from CQL in focussing on queries of linguistic structures, and separating document and span query concepts (see Section 3).

[1]CQLF is short for *Corpus Query Lingua Franca*, which is part of the ISO TC37 SC4 Working Group 6 (ISO/WD 24623-1).

[2]Like KoralQuery, XCQL is not meant to be human readable, but to represent query expressions as machine readable tree structures. For various compilers from CQL to XCQL, see `http://zing.z3950.org/cql/`; last accessed 27 April 2015.

3 Query Representation

KoralQuery is serialized to JSON-LD (Sporny et al., 2014), a JSON (Crockford, 2006) based format for Linked Data, which makes it possible for corpus query systems to interoperate by exchanging the common protocol.[3] JSON-LD relies on the definition of object types via the `@type` keyword, thus informing processing software of the attributes and values that a particular object may hold. As can be seen in the example serializations in this section (see Fig. 1-3), KoralQuery makes use of the `@type` keyword to declare query object types. Those types fall into different categories that we introduce in the remainder of this section.[4]

While KoralQuery aims to express as many different linguistic and metalinguistic query structures as possible, it currently guarantees to represent types and operations defined in *Poliqarp QL* (Przepiórkowski et al., 2004), *COSMAS II QL* (Bodmer, 1996) and *ANNIS QL* (Rosenfeld, 2010). In addition, the protocol comprises a subset of the elements of CQL (OASIS Standard, 2013).

As JSON-LD objects can reference further namespaces (via the `@context` attribute), KoralQuery is arbitrarily extensible.

3.1 Document Queries

KoralQuery allows to specify metadata constraints that act as filters for virtual collections using the `collection` attribute. Those metadata constraints, so-called **collection types**, serve a dual purpose: Besides the obvious benefit of allowing users to restrict their search via dynamic sampling to documents that meet specific requirements on metadata such as publication date, authorship or genre, they can be used to control access to texts that the user has no permission to read (cf. Sec. 3.3).

A single metadata constraint is called a **basic collection type**, and defines a metadata field, a value and a match modifier, for example to negate the constraint. Basic collection types can be combined using boolean operators (AND and OR) to recursively form **complex collection types**. The result of a collection type is a collection of documents which satisfy the encoded constraint (or combination of constraints), for instance all documents that were published after a certain date or that contain a certain string of characters in their title. Figure 1 illustrates the serialization of a simple virtual collection definition.

```
{
  "@context" : "http://korap.ids-mannheim.de/ns/
      koral/0.3/context.jsonld",
  "collection" : {
    "@type" : "koral:doc",
    "key" : "pubDate",
    "value" : "2005-05-25",
    "type" : "type:date",
    "match" : "match:geq"
  },
  "query" : {}
}
```

Figure 1: KoralQuery serialization for a virtual collection that is restricted to documents with a `pubDate` of greater or equal than `2005-05-25`.

3.2 Span Queries

To find occurrences of particular linguistic structures in corpus data (possibly restricted through the aforementioned document queries), KoralQuery uses the attribute `query`, under which it registers objects of specific, well-defined types. Those objects, along with their hierarchical organization, represent the linguistic query issued by the user.[5]

The intended generic usability of KoralQuery demands a high degree of flexibility in order to cover as many linguistic phenomena and theories as possible. It must therefore be maximally independent of, and neutral with regard to,

(i) the type and structure of linguistic annotation on the text data,

(ii) the choice of specific tag sets, e.g. for part-of-speech annotations or dependency labels.

KoralQuery achieves this neutrality by instantiating distinct linguistic types as abstract structures which can flexibly address different sources and layers of linguistic annotation at the same time. Linguistic patterns of greater complexity can be defined by using a modular system of nestable types and operations, drawing on various familiar search technologies and formalisms, includ-

[3]JSON-LD was chosen to be compatible with LAPPS recommendations from ISO TC37 SC4 WG1-EP, as suggested by Piotr Bański.

[4]The type categories are set in boldface. A detailed definition of types and attributes is provided by the KoralQuery specification (Diewald and Bingel, 2015), which may serve as a reference for implementers of KoralQuery processors.

[5]As the response format is not part of the KoralQuery specification, the result handling is subject to the query engine. It may, for instance, return surrounding text spans or the total number of occurrences.

ing concepts from regular expressions, XML tree traversal, boolean search and relational database queries.

The nesting principle of KoralQuery states that objects describing linguistic structures in the corpus data, so-called **span types**, may be embedded in parental objects to recursively describe complex linguistic structures, thus forming a single-rooted tree.

Span types may be further sub-classified into basic and complex types. **Basic span types** denote linguistic entities such as words, phrases and sentences that are annotated in the corpus data. The result of such a span type is a text span, which in turn is defined through a start and an end offset with respect to the primary text data. **Complex span types** define linguistic or result-modifying operations on a set of embedded span types, which thus act as arguments (or *operands*) of the relation and pass their resulting text spans on to the parent operation.[6] Such operations may express syntactic relations or positional constraints between spans.

Figure 2, for example, represents a span query of two `koral:token` objects (basic span types) each wrapping a single `koral:term` object, whose resulting text spans are required to be in a sequence (i.e. follow each other immediately in the order they appear in the list), as formulated by the `operation:sequence` in the embedding `koral:group` object (a complex span type).

Leaf objects of the span query tree structure may either be basic span types or **parametric types**, containing specific information that is requested for certain span types. They are intended to normalize the usage and representation of similar or equal parameters used across different types. The `koral:term` objects in Figure 2, which express constraints on their parent `koral:token` objects, are examples of such parametric types and are used to uniformly access annotation labels from different sources and on different layers. Next to such **basic parametric types**, KoralQuery provides **complex parametric types** that encode, for instance, logical operations on other parametric types (see the `koral:termGroup` in Figure 2).

Note that all of those types are themselves complex structures in that they are composed of a specific set of obligatory and optional attributes that carry corresponding values. Those values, in turn, are also constrained to be of specific data types. They can either be primitives (like string, integer or boolean), parametric KoralQuery types, or controlled values.

[6]In addition, the `koral:reference` type may refer to objects elsewhere in the tree, which provides a mechanism similar to ID/IDREF in XML. This strategy is necessary to support graph-based query structures found in certain query languages.

```
{
  "@context" : "http://korap.ids-mannheim.de/ns/
       koral/0.3/context.jsonld",
  "collection" : {},
  "query" : {
    "@type":"koral:group",
    "operation" : "operation:sequence",
    "operands" : [ {
      "@type" : "koral:token",
      "wrap" : {
        "@type" : "koral:termGroup",
        "relation" : "relation:and",
        "operands" : [ {
          "@type" : "koral:term",
          "foundry" : "tt",
          "key" : "ADJA",
          "layer" : "pos",
          "match" : "match:eq"
        }, {
          "@type" : "koral:term",
          "foundry" : "cnx",
          "key" : "@PREMOD",
          "layer" : "syn",
          "match" : "match:eq"
        } ]
    }, {
      "@type" : "koral:token",
      "wrap" : {
        "@type" : "koral:term",
        "key" : "octopus",
        "layer" : "lemma",
        "match" : "match:eq"
      }
    } ]
  }
}
```

Figure 2: KoralQuery serialization for a premodifying adjective followed by the lemma *octopus*. The dual constraint on the first token (adjective and premodifying) is reflected by the `koral:termGroup`, which expresses a conjunction of the two `koral:term` objects. The different values for `foundry` indicate that different annotation sources are addressed.

3.3 Query Rewrites

Query processors may perform a wide range of different tasks aside of searching. Examples include the modification of queries to restrict access to certain documents, to improve recall (e.g. by introducing synonyms or suggesting query reformulations), or to inject missing query elements (like

```
{
  "@context" : "http://korap.ids-mannheim.de/ns/
      koral/0.3/context.jsonld",
  "collection" : {
    "@type" : "koral:docGroup",
    "operation" : "operation:and",
    "operands" : [ {
      "@type" : "koral:doc",
      "key" : "pubDate",
      "value" : "2005-05-25",
      "type" : "type:date",
      "match" : "match:geq"
    }, {
      "@type" : "koral:doc",
      "key" : "corpusID",
      "value" : "Wikipedia",
      "rewrites" : [ {
        "@type" : "koral:rewrite",
        "src" : "Kustvakt",
        "operation" : "operation:injection"
      } ]
    } ]
  },
  "query" : {}
}
```

Figure 3: Rewritten KoralQuery instance (see Figure 1), with an injected access restriction.

preferred annotation tools) based on user settings (Bański et al., 2014). Queries may also be analyzed for the most commonly queried structures, for instance to perform query and index optimization or to shed light on which texts and annotations are most popular with the users. In a post-processing step, queries can also be transformed for visualization purposes, for example to illustrate sequences or alternatives in complex query structures.

Using a well-defined and widely adopted serialization format such as JSON makes it easy to perform such tasks, and KoralQuery supports this kind of pre- and post-processors even further by introducing mechanisms to trace query rewrites by using so-called **report types** that are passed to further processors in the processing pipeline. In this way, query modifications (like the aforementioned rewrites for access restriction and recall improvements) can be made visible and transparent to the user. In this respect, KoralQuery differs from common database query systems, where rewrites are internal and hidden from the user (Huey, 2014).

In Figure 3, the virtual collection of Figure 1 is rewritten by the processor *Kustvakt* in a way that a further constraint is injected, limiting the virtual collection to all documents with a `corpusID` of `Wikipedia` (i.e. excluding all documents from other corpora). This rewrite is documented by the `koral:rewrite` object (a report type). Documenting rewrites is optional (e.g. the injected `operation:and` in the example Figure is implicit and was not reported using `koral:rewrite`).

In addition, KoralQuery allows to report on various processing issues (independent of rewrites, e.g. regarding incompatibilities) by using the `errors`, `warnings`, and `messages` attributes.

Report types (in opposition to collection types, span types, and parametric types) do not alter the expected query result.

4 Implementations

KoralQuery is the core protocol used in KorAP[7] (Bański et al., 2013), a corpus analysis platform developed at the Institute for the German Language (IDS). KorAP is designed to handle very large corpora and to be sustainable with regard to future developments in corpus linguistic research. This is ensured through a modular architecture of interoperating software units that are easy to maintain, extend and replace. The interoperability of components in KorAP is certified through the use of KoralQuery for all internal communications.

Koral[8] translates queries from various corpus query languages (as mentioned in Section 3) to corresponding KoralQuery documents. This conversion is a two-stage process, which first parses the input query string using a context-free grammar and the ANTLR framework (Parr and Quong, 1995) before it translates the resulting parse tree to KoralQuery.

Krill[9] is a corpus search engine that expects KoralQuery instances as a request format. To index and retrieve primary data, textual annotations and metadata of documents as formulated by KoralQuery, Krill utilizes *Apache Lucene*.[10]

Kustvakt is a user and corpus policy management service that accepts KoralQuery requests and rewrites the query as a preprocessor (see Sec. 3.3) before it is passed to the search engine (e.g. Krill). Rewrites of the document query may restrict the requested collection to documents the user is allowed to access, while the span query may be modified by injecting user defined properties.

[7]`http://korap.ids-mannheim.de/`

[8]`http://github.com/KorAP/Koral`; Koral is free software, licensed under BSD-2.

[9]`http://github.com/KorAP/Krill`; Krill is free software, licensed under BSD-2.

[10]`http://lucene.apache.org/core/`

5 Summary and Further Work

We have presented KoralQuery, a general protocol for queries to linguistic corpora, which is serialized as JSON-LD. KoralQuery allows for a flexible representation and modification of corpus queries that is independent of pre-defined tag sets or annotation schemes. Those queries pertain to both selection of documents by metadata or content, and text span retrieval by the specification of linguistic patterns. To this end, the protocol defines a set of types and operations which can be nested to express complex linguistic structures. By employing an automatic conversion from several QLs to KoralQuery, corpus engines may allow their users to choose the QL that they are most comfortable with or that are best equipped to answer their research questions.

The KoralQuery specification (Diewald and Bingel, 2015) does not claim to be complete or to cover all possible linguistic types and structures. Amendments to the protocol may follow in future versions or may be implemented by individual projects, which is easily done by supplying an additional JSON-LD `@context` file that links new concepts to unique identifiers. Extensions that we consider for upcoming versions of KoralQuery include text string queries that are not constrained by token boundaries and more powerful stratification techniques for virtual collections.

Acknowledgements

KoralQuery, as well as the described implementation components, are developed as part of the KorAP project at the Institute for the German Language (IDS)[11] in Mannheim, member of the Leibniz-Gemeinschaft, and supported by the KobRA[12] project, funded by the Federal Ministry of Education and Research (BMBF), Germany. The authors would like to thank their colleagues for their valuable input.

References

Piotr Bański, Joachim Bingel, Nils Diewald, Elena Frick, Michael Hanl, Marc Kupietz, Piotr Pezik, Carsten Schnober, and Andreas Witt. 2013. KorAP: the new corpus analysis platform at IDS Mannheim. In Zygmunt Vetulani and Hans Uszkoreit, editors, *Human Language Technologies as a Challenge for Computer Science and Linguistics. Proceedings of the 6th Language and Technology Conference*, Poznań. Fundacja Uniwersytetu im. A. Mickiewicza.

Piotr Bański, Nils Diewald, Michael Hanl, Marc Kupietz, and Andreas Witt. 2014. Access Control by Query Rewriting: the Case of KorAP. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavik, Iceland, may. European Language Resources Association (ELRA).

Franck Bodmer. 1996. Aspekte der Abfragekomponente von COSMAS II. *LDV-INFO*, 8:142–155.

Douglas Crockford. 2006. The application/json Media Type for JavaScript Object Notation (JSON). Technical report, IETF, July. `http://www.ietf.org/rfc/rfc4627.txt`.

Nils Diewald and Joachim Bingel. 2015. KoralQuery 0.3. Technical report, IDS, Mannheim, Germany. Working draft, in preparation, `http://KorAP.github.io/Koral`, last accessed 27 April 2015.

Elena Frick, Carsten Schnober, and Piotr Bański. 2012. Evaluating query languages for a corpus processing system. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*, pages 2286–2294.

Patricia Huey, 2014. *Oracle Database, Security Guide, 11g Release 1 (11.1)*, chapter 7. Using Oracle Virtual Private Database to Control Data Access, pages 233–272. Oracle. `http://docs.oracle.com/cd/B28359_01/network.111/b28531.pdf`, last accessed 27 April 2015.

OASIS Standard. 2013. searchRetrieve: Part 5. CQL: The Contextual Query Language Version 1.0. `http://docs.oasis-open.org/search-ws/searchRetrieve/v1.0/os/part5-cql/searchRetrieve-v1.0-os-part5-cql.html`.

Terence J. Parr and Russell W. Quong. 1995. ANTLR: A predicated-LL (k) parser generator. *Software: Practice and Experience*, 25(7):789–810.

Adam Przepiórkowski, Zygmunt Krynicki, Lukasz Debowski, Marcin Wolinski, Daniel Janus, and Piotr Bański. 2004. A search tool for corpora with positional tagsets and ambiguities. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 1235–1238. European Language Resources Association (ELRA).

Viktor Rosenfeld. 2010. An implementation of the Annis 2 query language. Technical report, Humboldt-Universität zu Berlin.

Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. 2014. JSON-LD 1.0 – A JSON-based Serialization for Linked Data. Technical report, W3C. W3C Recommendation, `http://www.w3.org/TR/json-ld/`.

[11]`http://ids-mannheim.de/`

[12]`http://www.kobra.tu-dortmund.de/`

Reflections and a Proposal for a Query and Reporting Language for Richly Annotated Multiparallel Corpora

Simon Clematide
Institute of Computational Linguistics, University of Zurich
simon.clematide@cl.uzh.ch

Abstract

Large and open multiparallel corpora are a valuable resource for contrastive corpus linguists if the data is annotated and stored in a way that allows precise and flexible ad hoc searches. A linguistic query language should also support computational linguists in automated multilingual data mining. We review a broad range of approaches for linguistic query and reporting languages according to usability criteria such as expressibility, expressiveness, and efficiency. We propose an architecture that tries to strike the right balance to suit practical purposes.

1 Introduction

There is a large amount (millions of sentences) of open multiparallel text data available electronically: resolutions of the General Assembly of the United Nations (Rafalovitch and Dale, 2009), European parliament documents (Koehn, 2005; Hajlaoui et al., 2014), European administration translation memories and law texts (Steinberger et al., 2012; Steinberger et al., 2006), documents from the European Union Bookstore (Skadiņš et al., 2014), and movie subtitles. See Tiedemann (2012) and Steinberger et al. (2014) for an overview.

Automatic part-of-speech tagging and lemmatization of raw text has become standard procedure, and richer linguistic annotations such as morphological analysis, named entity recognition, base chunking, and dependency analysis are possible for many languages. Further, statistical word alignment can be applied to any parallel language resource. If we want to exploit these large, richly annotated resources and flexibly serve the language-related information needs of translators, terminologists and contrastive linguists, an expressive query language for ad hoc search must be provided. Such a query language will also be useful for automated *linguistic data mining*, a use case of computational linguists. A successful combination of these two different paradigms of linguistic information retrieval (i.e. ad hoc search and precomputed word collocation statistics) has been shown in the case of the text corpus query language CQL within the framework of the *Sketch Engine* (Kilgarriff et al., 2014).

Historically, there are two different strains of linguistic query systems, (a) corpus linguistics tools for text corpora such as CQP (Christ, 1994) with KWIC reporting, and (b) treebank tools such as TGrep2 (Rohde, 2005) for searching through deeply nested structures of syntactically annotated sentences. In recent years, we have seen a convergence of these strains: query languages for text corpora have enriched their search operators in order to cope with syntactic constituents, for example introducing the operators `within` and `contain` in CQL (Jakubicek et al., 2010) or the constituent search construct in Poliquarp (Janus and Przepiórkowski, 2007). On the other hand, treebanking-style query approaches that were bound to context-free tree structures have evolved into more general query systems for structural linguistic annotations, e.g. ANNIS (Krause and Zeldes, 2014) which allows a richer set of the structural relations (multi-layered directed acyclic graphs, including syntactic dependencies or coreference chains across sentences), or the Prague Markup Language Tree Query (PML-TQ) system for multi-layered annotations (Štěpánek and Pajas, 2010), which also covers parallel treebanks.[1]

[1] Unfortunately, it is difficult to access up-to-date information about the query possibilities for alignments of words or syntactic nodes. The documentation, however, describes a general cross-layer, node-identifier-based selector dimension. The parallel Prague Czech-English Dependency Treebank 2.0 (PCEDT 2.0) `http://ufal.mff.cuni.cz/pcedt2.0` illustrates the representation of word-aligned dependency trees.

1.1 Linguistic Information Needs

A *linguistic query* in a general sense is a set of interrelated constraints about linguistic structures. The following paragraphs introduce the structures we want to represent and query.

Monolingual constraints on the primary level of word tokens (the minimal unit of analysis) are dealing with inflected word forms, base forms, part-of-speech tags, and morphological categories. Word tokens have a sequential ordering relation (linear precedence). For our case of orthographically well-formed texts, we assume consistent tokenization for all levels of annotation. Giving up this requirement leads to non-trivial ordering problems (Chiarcos et al., 2009). Sentences are sequences of tokens, and documents are sequences of sentences.[2] Documents or sentences typically have metadata associated with them, for instance indicating whether a document is a translation or not.

Each full or partial dependency analysis of a sentence can be represented as a directed and labeled tree graph where each node is a word token, except for the root of the tree, which we assume to be a virtual node. Nested syntactic constituents (or chunks in the case of partial parsing) introduce a dominance relation between syntactic nodes (non-terminals) or primary token nodes (terminals). Dominated nodes also have a linear precedence ordering, the sibling relation.

Cross-lingual constraints are concerned with word alignments and sub-sentential alignments on the chunk level.[3] Directed bilingual word alignments as produced by statistical word alignment tools such as GIZA++ are 1:*n* (Och and Ney, 2003). Bidirectional alignments are thus relational, in general, we have $m : n$ alignments on the level of words, for example, between a German compound and its corresponding multi-word unit in French, unless we apply a symmetrization technique (Tiedemann, 2011, 75ff.).[4]

1.2 Reporting and Visualization

The set of constraints in a query does not exactly determine the content or format of the search results. All flexible linguistic query languages offer means to select the sub-structures and attributes which the user is interested in.[5] This may also include sorting, aggregating or statistical tabulating of the results, as for instance the excellent reporting functions of PML-TQ allow. In our opinion, reporting also includes the user-configurable export of search results, for example as simple comma-separated data for further statistical processing[6], or as hierarchically structured XML serializations.

The graphical visualization of search results aids end users in quickly browsing complex data structures. Visualizations of syntactic structures or frequency distributions of aligned words should be generated on top of specific textual reporting formats. Interactive behavior (collapsing trees, highlighting of aligned nodes) supports a quick interpretation of search results.

The remainder of this paper is structured as follows. Section 2 describes general usability criteria of linguistic query systems. Section 3 discusses interesting linguistic query languages and their main properties. Section 4 introduces general data query languages that are related to linguistic systems. Section 5 discusses evaluation approaches for linguistic query languages. Finally, section 6 presents our proposals for an efficient linguistic query and reporting system for multiparallel data.

2 Usability Criteria for Linguistic Query Systems

Expressibility How naturally can users express their information need? Can users apply their linguistic concepts to formulate their query (Jakubicek et al., 2010, 743), or do they have to deal with cumbersome constructs?

Non-experts may profit from a visual or menu-based composition of queries. Gärtner et al. (2013) and Mírovský (2008) describe graphical query solutions for dependency trees. ANNIS (Zeldes et al., 2009) offers a graphical query interface for AQL. Nygaard and Johannessen (2004) built a menu-based visual query composition for parallel treebanks that used TGrep2 as its query execution engine.

[2]In order to keep the description simple we do not impose more nesting levels in documents.

[3]Sentence alignments are considered as given in the context of multiparallel corpora, although in practical terms it might require a lot of work to achieve a proper and consistent sentence alignment across multiple languages.

[4]Recently, Baisa et al. (2014) applied Dice coefficients to identify aligned lemmas in parallel sentences.

[5]TGrep2 uses backticks to mark the top node of the subtree that is printed as output.

[6]ANNIS provides a practical export format for the WEKA machine learning framework.

Experts, however, will profit most from text-based queries that allows to abstract common and recurrent functionality in the form of user-definable macros, variables, or functions.

Expressiveness Are there inherent limitations in a query language that systematically prevent the formulation of precise search constraints for certain structures? It is well known since its inception that the fragment of existential first-order logic implemented by the TIGERSearch language does not allow for the search of missing constituents in syntactic graphs (König and Lezius, 2003). Lai and Bird (2010) provide a concise overview on the formal expressiveness of query languages for hierarchical linguistic structures and discuss the fact that transitive closures of immediate dominance or precedence relations formally require the expressiveness of monadic second-order logic. Interestingly, such a high expressiveness does not imply inefficient or impractical execution times as shown by Maryns and Kepser (2009) for context-free treebank structures – if tree automata techniques are used. However, purely logical approaches have not received much attention in practice.

Efficiency How much processing time and memory is needed for the execution of a query? Answers to this question relate to many different parameters. First, *data size* of the corpora matters – dealing with thousands, millions, or billions of sentences makes a big difference. Second, *data model complexity* matters. Third, *query expressiveness and complexity* matters.

Even if a user is dealing with large datasets, complex data models and complicated queries, there are solutions to produce acceptable response times. For instance, by providing a highly parallel computing infrastructure using MapReduce techniques (Schneider, 2013), or by using sophisticated indexing and retrieval techniques (Ghodke and Bird, 2012).

Reporting and exporting Does the query language or query system offer flexible support for the user to configure the data reported in the search results? The selection of sub-structures is typically deeply integrated in the query syntax. For text concordancing tools, Frick et al. (2012) mention the `LINK/ALL` operator of COSMAS II, or bracketed expressions in Poliquarp. The statistical reporting functions of the monolingual treebank search tool TIGERSearch[7] rely on named node specifications, and they can only be accessed and configured by graphical user interface interactions. Other query languages such as PML-TQ offer a proper reporting language with a rich set of functions for sorting, aggregating and exporting (e.g. grammar rules).

Visualization Does the query system offer appealing visualizations of the data or data aggregations? ANNIS3 (Krause and Zeldes, 2014) has an outstanding amount of visualization options.

Availability and accessibility Is a system bound to specific operating systems? Large datasets typically overstrain personal desktop computers. Web-based services can be hosted on dedicated computing infrastructure, and there is typically no client-side software installation necessary given the rendering capabilities of modern web browsers (e.g. interactive SVG graphics). Open web-based services enable easy sharing of query results via URLs (Pezik, 2011).

3 Families of Linguistic Query Languages

As mentioned above, there are two strains of linguistic query languages. Some specific properties of these languages are discussed next.

3.1 Text Corpus Query Languages

CQP The language of the IMS Corpus Query Processing Workbench (Hardie, 2012)[8] has a long history (Christ, 1994). From this common ancestor, CQL (Kilgarriff et al., 2004) and Poliquarp were later developed. Right from the beginning, CQP supported annotated word tokens, structural boundaries (sentences, constituents) and sentence-aligned parallel texts. The core of a query consists of regular expressions that specify matching token sequences. These descriptions can refer to the level of word forms, part-of-speech tags or any other positional (=token-bound) attribute. Non-recursive constituents are indirectly available as structural boundaries and can be used to restrict the search space for regular expression matches on the positional level. The constituent segments also allow for attributes which can be queried, for instance syntactic head information. The main

[7]`http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/tigersearch.html`

[8]`http://cwb.sourceforge.net`

Relation	QL	Symbol
Immediate dominance	TGrep2, fsq, TS, AQL	`>`
	LPath	`/`
Transitive dominance	TGrep2	`>>`
	fsq	`>+`
	TS, AQL	`>*`
	LPath	`//`
Immediate precedence	TGrep2, fsq, TS, AQL	`.`
	LPath	`->`
Transitive precedence	TGrep2, fsq	`..`
	TS, AQL	`.*`
	LPath	`-->`
Immediate sibling	TS, AQL	`$`
	TGrep2	`$.`
	LPath	`=>`

Table 1: Operators of query languages (QL)

weakness of this query language is the lack of a means to query arbitrary relations between tokens, which would be necessary to properly support the search for dependency relations. Given the fact that dependency labels are bound to words, one could map this information as an attribute on the positional level, for example, attributing the property of being a subject to the head of the subject.

An integrated macro and reporting language distinguishes CQP as a powerful and versatile tool.

CQL The query language behind the commercial corpus query platform Sketch Engine[9] is an extension of CQP (Jakubicek et al., 2010).

Support for identifying word matches across parallel corpora is technically implemented via the `within` operator. For a sentence-aligned parallel corpus (English and German Europarl corpus), a query rooted in the English side might look like:

```
[word="car"] within europarl7_de: [word="Auto"]
```

This finds all occurrences of *car* in sentences where a parallel sentence containing the word *Auto* exists. This kind of query, however, does not allow to explicitly test for word alignment relations. Still, the search patterns on both sides of the `within` operator can be arbitrarily complex.

3.2 Treebank Query Languages

TGrep2 The efficient treebank query tool TGrep2 is limited to context-free parse trees. Lai and Bird (2004) see its strength in the ability to query for non-inclusion or non-existence of constituents. Their information need Q2 "Find sentences that do not include the word *saw*" can be expressed succinctly as `S !<< saw`. Their information need Q5 "Find the first common ancestor of sequences of a noun phrase followed by a verb phrase" leads to a short but intricate query (see Tab. 1 for operators):

```
*=p << (NP=n .. (VP=v >> =p !>>
                 (* << =n >> =p)))
```

3.2.1 Path-based Languages

LPath Bird et al. (2006) developed this query language as a generally applicable extension of the XPath query language for XML[10]. Syntactic trees as well as XML documents are ordered trees. However, the direct use of XPath for querying linguistic trees is limited by the absence of (a) the horizontal axis of *x immediately follows/precedes y*, and (b) *sibling x immediately follows/precedes sibling y*.[11] Q2 from above can be stated as

```
/S[not //_[@lex = 'saw']]
```

Q5 cannot be expressed correctly (Lai and Bird, 2004). A further extension of LPath, called LPath+ (Lai and Bird, 2005), is more expressive and allows for a correct but complex query:

```
//_[/_[(NP or (/_[not(=>_)])*/NP[not(=>_)) and
      => (VP or (/_[not(<=_)])*/VP[not(<=_)])]
```

This is due to the fact that path-based, variable-free languages cannot easily express equality restrictions. Therefore, the following shorter LPath expression does not have the correct meaning because each NP (or VP) may refer to different nodes:

```
//_[{//NP->VP} and not(//_{//NP->VP})]
```

DDDQuery This language is another attempt to extend XPath and to better adapt it for linguistic information needs (Faulstich et al., 2006). Its data model was developed for a multi-layered, linguistically richly annotated representation of historical texts, including transcriptions and aligned translations, which resulted in "non-tree-shaped annotation graphs and multiple annotation hierarchies with conflicting structure". This query language "goes beyond LPath by supporting queries on text spans, on multiple annotation layers, and across aligned texts". The language introduces shared variables for any node set in order to easily express equality restrictions and report the matched nodes as result data.

[9]See Kilgarriff et al. (2014) for a recent description. The *NoSketchEngine*, the open-source part of the Sketch Engine, is available from `http://nlp.fi.muni.cz/trac/noske`.

[10]`http://www.w3.org/tr/xpath`

[11]Note that the transitive closures of these relations are available in XPath.

PML-TQ This query language is also a path-based approach (Štěpánek and Pajas, 2010). A query consists of a Boolean combination of node selector paths and filters. The language allows recursive sub-queries in selectors which evaluate to node sets. The cardinality of these node sets can be tested by numeric quantifiers. A quantifier of zero tests for the non-existence of nodes; therefore, non-existing nodes can be queried in a natural way. A similar technique of extensionalization of sub-queries into node sets was implemented for the TreeAligner language (Marek et al., 2008).

3.2.2 Logic-based Languages

fsq[12] The *Finite Structure Query* language (Kepser, 2003) provides full first-order logic as a query language over syntactic structures of the TIGER data model (Brants et al., 2004). This includes labelled secondary edges between arbitrary nodes and discontiguous children. Therefore, fsq has an outstanding expressiveness. Regular expression support for node labels and response times that are comparable to TIGERSearch make this approach a practical one. Lai and Bird's difficult question Q5 can be expressed as follows in the somewhat inconvenient LISP-like prefix notation for first-order logic of fsq [13]:

```
(E a (E n (E v (&
   (cat n NP) (cat v VP) (>+ a v) (.. n v)
   (! (>+ n v)) (! (>+ v n))
   (A b (-> (& (>+ a b) (>+ b n))
            (! (>+ b v))))))))
```

Compared to the query language of TIGERSearch, there is a lack of special purpose predicates such as the (token) arity of syntactic nodes or precedence or dominance restrictions with numeric distance limits, for example, `>2,5` expressing a indirect dominance relation with a minimal depth of 2 and a maximum of 5.

MonaSearch[14] Maryns and Kepser (2009) extended the logical expressiveness of fsq even further to monadic second-order logic. However, its data model is restricted to context-free parse trees. A main application of such an expressive language are automatic consistency checks in human-created treebanks. However, existentially quantified formulas can be used to effectively query matching structures.

[12]The Java implementation of fsq also includes a TIGERSearch-like visualization for the matched trees, see `http://www.tcl-sfs.uni-tuebingen.de/fsq`.

[13]Existential (`E`) and universal (`A`) quantification, conjunction (`&`), negation (`!`), implication (`->`).

[14]`http://www.tcl-sfs.uni-tuebingen.de/MonaSearch`

TIGERSearch König and Lezius (2000) introduced this logic-based, syntax graph description language for the TIGER data model. It is a subset of first-order logic, providing only globally existentially quantified variables and limited negation. The language has two layers, namely, node constraints and graph constraints.

Node constraints are either node descriptions or node (relation) predicates. *Node descriptions* are Boolean expressions of feature-value constraints with optional variable decorations for referencing the same node several times in a query, for instance `#v:[word != "saw"]` for a terminal node description, or `#np:[cat = ("NP"|"CNP")]` for a simple or coordinated noun phrase. *Node predicates* constrain selected properties of nodes, such as being the root of a tree (`root(#s)`) or having a certain number of daughter nodes (`arity(#CNP,2)`). *Node relation predicates* express the usual structural relations in a user-friendly operator notation, e.g. `#s >* #np` for a dominance relation. *Graph constraints* are conjunctions or disjunctions of node constraints. Negation is not allowed on the level of graph constraints, which severely limits the expressiveness.

The TIGER language originally specified user-defined macros (templates), however, this part of the language was never implemented.

AQL The query language of ANNIS is an extension of the TIGERSearch language for multi-level graph-based annotations. It offers operators for labelled dependency relations, inclusion or overlap of token spans, corpus metadata information, and namespaces for annotations of the same type produced by different tools[15]. The operator for dependency relations is an instance of the general operator `->` for directed and labelled edges between any two nodes. Such edges can also be used to establish or query alignments between parallel sentences on the level of words or phrases.

TreeAligner The *Stockholm TreeAligner* (Lundborg et al., 2007) introduced an operator for querying bilingual alignments between words or phrases of parallel treebanks, freely combinable with monolingual TIGERSearch-style queries. To overcome some expressiveness limitations of

[15]For instance, for different parsers (Chiarcos et al., 2010).

TIGERSearch, Marek et al. (2008) introduced node sets (node descriptions decorated with variables starting with % instead of #). One might try to express Bird and Lai's Q2, that is, find sentences without *saw*, in the following ways:

```
#s:[cat="S"] >* #w:[word!="saw"]   (1)
#s:[cat="S"] !>* #w:[word="saw"]   (2)
#s:[cat="S"] !>* %w:[word="saw"]   (3)
```

(1) actually matches all cases where a sentence dominates any other word than *saw*. (2) searches for occurrences of the word *saw* not dominated by a sentence node. The interpretation of (3) relies on a modified evaluation strategy of the negated dominance if one of the arguments is a node set: only those sentences match where the negated transitive dominance constraint `!>*` is true for any of the nodes with the word attribute *saw*.

4 General Data Query Languages

Complex data structures are not a privilege of linguistics, so obviously many *general* data query languages and data management systems exist. Some of them have been used to represent and query linguistic structures.

XPath/XQuery[16] Bouma and Kloosterman (2007) used these XML technologies in a straightforward manner for querying and mining syntactically annotated corpora. These query languages are also the basis of Nite QL (Carletta et al., 2005), which is targeted at multimodal annotations.

SQL The structured query language for relational databases (RDBMS) is a standard technology with highly efficient implementations. RDBMSs have been widely used to represent large amounts of data, e.g. for text concordancing.[17]

CYPHER[18] Distributed NoSQL graph databases and CYPHER as one of the straightforward query languages seem to be a good match for highly interconnected linguistic data (Holzschuher and Peinl, 2013). Pezik (2013) reports some experiments for corpus representation and corpus query with a pure graph database. Banski et al. (2013) integrate a general text retrieval engine with a graph database for their corpus analysis platform.

[16]`http://www.w3.org/XML/Query`

[17]`http://corpus.byu.edu` (Davies, 2005)

[18]`http://neo4j.com/developer/cypher-query-language`

SPARQL[19] RDF (Resource Description Framework) triple stores with SPARQL endpoints for querying linked data are fairly standard nowadays. Kouylekov and Oepen (2014) used this technique to represent and query semantic dependencies. However, the queries directly operate on the internal RDF representations and do not meet the criteria of natural expressibility. The authors propose a query-by-example and a template expansion front-end for better usability.

Chiarcos (2012) introduce POWLA, a generic formalism to represent multi-layer annotated corpora in RDF and OWL/DL and to query these structures by SPARQL. In order to improve the expressibility, SPARQL macros for AQL operators are defined. Given the expressiveness of SPARQL, this allows to overcome the query language limitations of AQL or TIGERSearch, which cannot query for missing annotations.

LUCENE[20] Every information retrieval system has an integrated query language. Powerful text indexing and query engines such as LUCENE can be used to manage large amounts of texts. By treating each sentence as an IR document, Ghodke and Bird (2012) implemented a high-performance treebank query system[21] on top of LUCENE.

5 Evaluation Strategies for Linguistic Query Languages

There are essentially two approaches to implement the evaluation of linguistic query languages: either by programming a custom implementation of the execution of the query over a custom implementation of the data management, or by translating the query and the data into a host database system and executing the actual query on the host.

Sometimes, these approaches are mixed; for instance, the TreeAligner uses the relational database SQLite for storing and retrieving the primary data of word tokens, but implements a custom in-memory engine for the evaluation of the Boolean algebra of node predicates and node relations.

5.1 Custom Evaluation Engines

Manatee (Rychlý, 2007) is CQL's back-end for textual data management and query evaluation. It

[19]`http://www.w3.org/TR/sparql11-query`

[20]`http://lucene.apache.org`

[21]Their query language, however, does not allow regular expressions over labels, or underspecified node descriptions.

is language and annotation independent and includes efficient implementations of inverted indexes, word compression, etc. in order to cope with extremely large text corpora. Attributes of primary data can be set-valued and support unification-style attribute comparisons. Another interesting feature of Manatee is its support for dynamic attributes of positional primary data. These are implemented as function calls which can be declared at the level of the corpus configuration, for instance, for external lexicon look-up, morphological analysis, or the transformation of tags.

TGrep2, TIGERSearch and fsq are examples for treebank query systems with a fully custom data management and query evaluation engine. Rosenfeld (2010) gives a concise description of the implementation techniques behind TIGERSearch. The corpus import of TIGERSearch includes the construction of many specialized indexes for predicates and attributes. During indexing, statistics on the selectivity of attributes are built, which in turn guide the query execution planner to limit the full evaluation of a query to a subset of syntactic trees. At the stage of corpus indexing, users can provide their own type definitions, that is, short names for subsets of admissible feature values. A definition for genitive or dative case looks as follows:

```
    gen-dat := "gen","dat";
```

Although any query involving this case ambiguity can be expressed by a Boolean disjunction, type definitions lead to both more readable and compact queries and also to more efficient processing due to the type-based data model of TIGERSearch.

5.2 Query Translation Approach

LPath and DDDQuery are both Xpath-style languages that owe much to the hierarchical data model of XML. However, storing and efficient retrieval of large XML data sets turns out to be a technical challenge in general (Grust et al., 2004). One common solution for high-performance XML retrieval is based on a mapping of the hierarchical document structure into a flat relational format, which in turn allows the use of highly efficient RDBMSs. Both linguistic query languages – LPath and DDDQuery – are translated into SQL queries because their XML data model is physically stored in an RDBMS. The implementation of DDDQuery (Faulstich et al., 2006) is especially interesting for us because they first translate into a first-order logic intermediate representation from which the actual SQL queries are derived.

The development of the relational data model of ANNIS (relANNIS) and the corresponding translation of the ANNIS query language AQL into SQL queries by Rosenfeld (2010) was inspired by the DDDQuery translation. In the next section, we propose a linguistic query language which is similar to AQL but has a simpler data model. Therefore, we expect that our query translation component can be built using the techniques of AQL query evaluations.

6 A Proposal for Querying Richly Annotated Multiparallel Text Corpora

Our data model presupposes the following components: (a) multiparallel corpora with sentence, word and sub-sentential alignments across languages; (b) monolingual linguistic annotations such as PoS tags (preferably the same universal tagset across languages), base forms and morphological information; (c) syntactic annotations in the form of dependency relations and (partial) constituents, allowing the output of different tools for the same kind of analysis (multi-annotation). Multi-tokenization is not required for our data and would impose unnecessary complexity for the query component. However, metadata on the level of corpora, documents, or sentences is needed.

The proposed query language should allow to flexibly query all aspects of our data model. However, the search space of the query evaluation will be restricted to the context of a monolingual sentence and its corresponding aligned sentences.[22] The concrete query syntax for monolingual search will be based on TIGERSearch. Additionally, we introduce an alignment operator similar to the bilingual one of the TreeAligner. However, in multiparallel queries the alignment operator can be used to constrain alignments between nodes of any pair of languages. From AQL, we reuse the operator for dependency relations, the support for metadata predicates, and explicit namespaces. From CQP, we import the concept of a non-recursive macro language. Such a facility proved to be extremely useful for large scale linguistic mining in the case of Sketch Grammars of the Sketch Engine (Kilgarriff et al., 2004).

[22]Monolingual searches across sentence boundaries as permitted in CQP-style queries will not be possible. However, this search limit does not preclude reporting contextual information from surrounding sentences.

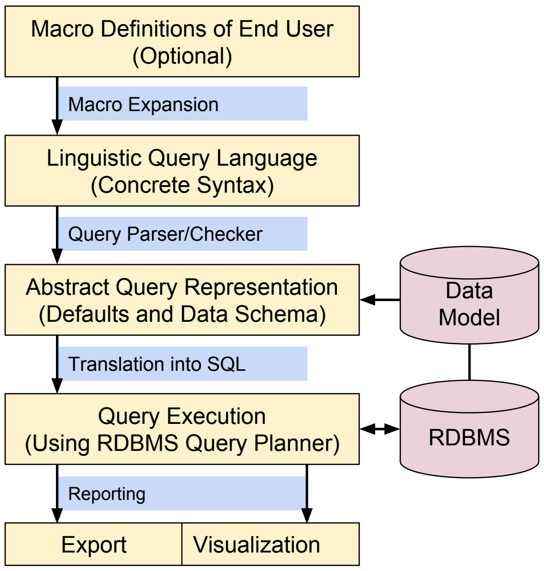


Figure 1: Architecture of our proposed system

The predicates needed for expressing the constraints of linguistic queries are different from the reporting functions. After the query execution, reporting functions will be applied to the token IDs, for instance the function *lemma*(*#wordid*) which renders the base form of a terminal node. Flexible reporting expressions similar to PML-TQ have to be defined and implemented. Graphical visualization is just another post-processing step that renders the output of specialized reporting functions.

RDBMSs are stable and efficient data management platforms, and modern, open source implementations such as PostgreSQL[23] support extensions to cope with acyclic graph structures (e.g. recursive SELECT). Therefore, we decided to host our data on an RDBMS and compile our linguistic query language into SQL. The overall architecture of our system is shown in Fig. 1.

One remaining problem is the inability to search for missing elements. The work presented here is part of a contrastive corpus linguistics project which is interested in differences in the use of articles in English and other languages, especially in the case where one language has an article and the other has not. A direct reimplementation of the TreeAligner approach with node set variables seems problematic since the evaluation of a query in the TreeAligner is implemented by iteratively constructing and manipulating node sets in memory. However, the general idea of an extensionalization of intermediate search results is natural.[24] Indeed, SQL itself offers the set operations UNION, INTERSECT, and EXCEPT to combine the results of different queries. In the next section, we present a proposal for searching missing elements using the result set operation EXCEPT.

6.1 Proposal for Query Result Set Operations

If we carefully separate reporting from querying, we can apply result set operations in order to implement the search for missing structures as filtering. We admit that there will be some computing overhead, but conceptually, filtering is easier for end users than full first-order logic.

To illustrate the idea, we informally embed CQP-style macros and TreeAligner constraints into SQL syntax. Bird and Lai's Q2 is easy:

```
SELECT #s FROM corpus WHERE #s:[cat="S"]
EXCEPT
SELECT #s FROM corpus
   WHERE #s:[cat="S"] >* [word="saw"]
```

The information need of Q5 focuses on a triple of an ancestor *a*, an NP *n* and a VP *v*.

```
MACRO a_dom_n_and_v($0=#a,$1=#n,$2=#v)
$0:[] >* $1:[cat="NP"] & $0 >* $2:[cat="VP"] &
$1 .* $2 & $1 !>* $2 & $2 !>* $1 ;

SELECT #a,#v,#n FROM corpus
   WHERE a_dom_n_and_v[#a,#n,#v]
EXCEPT
SELECT #a,#v,#n FROM corpus
   WHERE a_dom_n_and_v[#x,#n,#v] & #a >* #x
```

The first select is too general and includes all ancestors. The second selects the ancestors which dominate such an ancestor. The EXCEPT operator (which calculates the set minus) leaves the very ancestor that does not dominate any other.

A bilingual use case is the search for English noun chunks (nc) *without* article that are aligned to a German chunk *with* an article.[25] The information need are the parallel nouns.

```
MACRO aligned_nc($0=#c,$1=#n,$2=#c2,$3=#n2)
$0:[cat="NC"] > $1:[pos="NOUN"] &
$2:[cat="NC"] > $3:[pos="NOUN"] &
$1 --en,de $3 ;

SELECT #n_en,#n_de FROM corpus
  WHERE aligned_nc[#c_en,#n_en,#c_de,#n_de]
  & #c_de > [pos="DET"]
EXCEPT
SELECT #n_en,#n_de FROM corpus
  WHERE aligned_nc[#c_en,#n_en,#c_de,#n_de]
  & #c_de > [pos="DET"] & #c_en > [pos="DET"]
```

[23]`http://www.postgresql.org`

[24]Sub-selectors in PML-TQ work in a similar way and their quantifiers are cardinality tests on the matched node sets.

[25]We extend the alignment operator `A -- B` of the TreeAligner with language specifications `A --L1,L2 B`.

In all examples shown above, the resulting nodes of the combined SELECT statements will be fed into the desired reporting functions.

7 Conclusion

We provided a thorough review of linguistic query languages and their implementation approaches and tried to connect them to our use case of richly annotated multiparallel corpora. The usability of linguistic query systems is determined by their expressibility, expressiveness, and efficiency, their support for flexible reporting and exporting – including output for visualization back-ends – and open availability. We decided to host our highly structured data on a RDBMS and to provide a translation from a logic-based query language into SQL.

An important open question for future work is an empirical assessment whether our approach can efficiently deal with the huge amount of annotations and textual material of large multiparallel corpora. Furthermore, a performance comparison with approaches based on RDF triple stores and SPARQL (Chiarcos, 2012) is needed.

Acknowledgments

I wish to thank Johannes Graën for fruitful discussions and Rahel Oppliger for proofreading. This research was supported by the Swiss National Science Foundation under grant 105215 146781/1 through the project "Large-scale Annotation and Alignment of Parallel Corpora for the Investigation of Linguistic Variation".

References

Vít Baisa, Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, and Pavel Rychly. 2014. Bilingual word sketches: the translate button. In *Proc EURALEX*, pages 505–513.

Piotr Banski, J Bingel, N Diewald, E Frick, M Hanl, M Kupietz, P Pezik, C Schnober, and A Witt. 2013. KorAP: the new corpus analysis platform at IDS Mannheim. In *Human Language Technologies as a Challenge for Computer Science and Linguistics. Proceedings of the 6th Language and Technology Conference*.

Steven Bird, Yi Chen, Susan B. Davidson, Haejoong Lee, and Yifeng Zheng. 2006. Designing and evaluating an XPath dialect for linguistic queries. In *Proceedings of the 22nd International Conference on Data Engineering*, pages 52–.

Gosse Bouma and Geert Kloosterman. 2007. Mining syntactically annotated corpora with XQuery. In *Proceedings of the Linguistic Annotation Workshop*, pages 17–24.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Jean Carletta, Stefan Evert, Ulrich Heid, and Jonathan Kilgour. 2005. The NITE XML toolkit: Data model and query language. *Language Resources and Evaluation*, 39(4):313–334.

Christian Chiarcos, Julia Ritz, and Manfred Stede. 2009. By all these lovely tokens... merging conflicting tokenizations. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 35–43.

Christian Chiarcos, Kerstin Eckart, and Julia Ritz. 2010. Creating and exploiting a resource of parallel parses. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 166–171.

Christian Chiarcos. 2012. A generic formalism to represent linguistic corpora in RDF and OWL/DL. In *Proc LREC 2012*, pages 3205–3212.

Oliver Christ. 1994. A modular and flexible architecture for an integrated corpus query system. In *Proceedings of COMPLEX'95 3rd Conference on Computational Lexicography and Text Research Budapest, Hungary*, pages 23–32.

Mark Davies. 2005. The advantage of using relational databases for large corpora: Speed, advanced queries and unlimited annotation. *International Journal of Corpus Linguistics*, 10(3):307–334.

Lukas Faulstich, Ulf Leser, and Thorsten Vitt. 2006. Implementing a linguistic query language for historic texts. In *Current Trends in Database Technology*, pages 601–612.

Elena Frick, Carsten Schnober, and Piotr Bański. 2012. Evaluating query languages for a corpus processing system. In *Proc LREC 2012*, pages 2286–2294.

Sumukh Ghodke and Steven Bird. 2012. Fangorn: A system for querying very large treebanks. In *Proceedings of COLING 2012: Demonstration Papers*, pages 175–182, December.

Torsten Grust, Maurice Van Keulen, and Jens Teubner. 2004. Accelerating XPath evaluation in any RDBMS. *ACM Trans. Database Syst.*, 29(1):91–131, March.

Markus Gärtner, Gregor Thiele, Wolfgang Seeker, Anders Björkelund, and Jonas Kuhn. 2013. ICARUS – an extensible graphical search tool for dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Najeh Hajlaoui, David Kolovratnik, Jaakko Väyrynen, Ralf Steinberger, and Daniel Varga. 2014. DCEP - digital corpus of the European Parliament. In *Proc of LREC 2014*, pages 3164–3171.

Andrew Hardie. 2012. CQPweb — combining power, flexibility and usability in a corpus analysis tool. *International Journal of Corpus Linguistics*, 17(3):380–409.

Florian Holzschuher and René Peinl. 2013. Performance of graph query languages: Comparison of Cypher, Gremlin and native access in Neo4J. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204.

Milos Jakubicek, Adam Kilgarriff, Diana McCarthy, and Pavel Rychly. 2010. Fast syntactic searching in very large corpora for many languages. In *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 741–747.

Daniel Janus and Adam Przepiórkowski. 2007. Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 85–88.

Stephan Kepser. 2003. Finite structure query: A tool for querying syntactically annotated corpora. In *Proceedings of EACL*, pages 179–186.

Adam Kilgarriff, Pavel Rychlý, Pavel Smrž, and David Tugwell. 2004. The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pages 105–116.

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychly, and Vít Suchomel. 2014. The Sketch Engine: ten years on. *Lexicography*, 1(1):7–36.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit*, volume 5, pages 79–86.

Milen Kouylekov and Stephan Oepen. 2014. Semantic technologies for querying linguistic annotations: An experiment focusing on graph-structured data. In *Proc LREC 2014*, pages 4331–4336.

Thomas Krause and Amir Zeldes. 2014. Annis3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*.

Esther König and Wolfgang Lezius. 2000. A description language for syntactically annotated corpora. In *COLING 2000, July 31 - August 4, 2000, Universität des Saarlandes, Saarbrücken, Germany*, pages 1056–1060.

Esther König and Wolfgang Lezius. 2003. The TIGER language. A description language for syntax graphs. formal definition. Technical report, Institute for Natural Language Processing, University of Stuttgart.

Catherine Lai and Steven Bird. 2004. Querying and updating treebanks: A critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 139–146.

Catherine Lai and S. G. Bird. 2005. LPath+: A first-order complete language for linguistic tree query. In *Proc PACLIC'19*, pages 1–12.

Catherine Lai and Steven Bird. 2010. Querying linguistic trees. *J. of Logic, Lang. and Inf.*, 19(1):53–73, January.

Joakim Lundborg, Torsten Marek, Maël Mettler, and Martin Volk. 2007. Using the Stockholm TreeAligner. In *Proceedings of the 6th Workshop on Treebanks and Linguistic Theories*, pages 73–78.

Torsten Marek, Joakim Lundborg, and Martin Volk. 2008. Extending the TIGER query language with universal quantification. In *KONVENS 2008: 9. Konferenz zur Verarbeitung natürlicher Sprache*, pages 5–17.

Hendrik Maryns and Stephan Kepser. 2009. Monasearch - a tool for querying linguistic treebanks. In *Treebanks and Linguistic Theories 2009*, pages 29–40.

Jirí Mírovský. 2008. Netgraph - making searching in treebanks easy. In *In Proc. of IJCNLP'08*, pages 945–950.

Lars Nygaard and J.B. Johannessen. 2004. Searchtree - a user-friendly treebank search interface. In *Proc TLT 2004, Tübingen, December 10–11, 2004*, pages 183–189.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

Piotr Pezik. 2011. Providing corpus feedback for translators with the PELCRA search engine for NKJP. In *Explorations across languages and corpora : PALC 2009*, Łódź Studies in Linguistics, pages 135–144, Frankfurt am Main; New York. Peter Lang.

Piotr Pezik. 2013. Indexed graph databases for querying rich TEI annotation. http://digilab2.let.uniroma1.it/teiconf2013/wp-content/uploads/2013/09/Pezik.pdf.

Alexandre Rafalovitch and Robert Dale. 2009. United nations general assembly resolutions: A six-language parallel corpus. In *Proceedings of the MT Summit*, pages 292–299.

Douglas L. T. Rohde. 2005. TGrep2 user manual. http://tedlab.mit.edu/ dr/Tgrep2/tgrep2.pdf.

Viktor Rosenfeld. 2010. *An Implementation of the Annis 2 Query Language*. Student thesis, Humboldt-Universität zu Berlin.

Pavel Rychlý. 2007. Manatee/Bonito – a modular corpus manager. In *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN 2007*, pages 65–70.

Roman Schneider. 2013. KoGra-DB: Using MapReduce for language corpora. In *43. Jahrestagung der Gesellschaft für Informatik (GI)*, pages 140–142.

Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the EU Bookshop Corpus. In *Proc of LREC 2014*, pages 1850–1855.

Ralf Steinberger, B. Pouliquen, A. Widiger, C. Ignat, T. Erjavec, D. Tufiş, and D. Varga. 2006. The JRC-acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proc of LREC 2006*.

Ralf Steinberger, Andreas Eisele, Szymon Klocek, Spyridon Pilos, and Patrick Schlüter. 2012. DGT-TM: A freely available Translation Memory in 22 languages. In *Proc of LREC 2012*, pages 454–459.

Ralf Steinberger, Mohamed Ebrahim, Alexandros Poulis, Manuel Carrasco-Benitez, Patrick Schlüter, Marek Przybyszewski, and Signe Gilbro. 2014. An overview of the European Union's highly multilingual parallel corpora. *Language Resources and Evaluation*, 48(4):679–707.

Jörg Tiedemann. 2011. *Bitext Alignment*, volume 4 of *Synthesis Lectures on Human Language Technologies*. Morgan & Claypool.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proc of LREC 2012*, pages 2214–2218.

Amir Zeldes, Anke Lüdeling, Julia Ritz, and Christian Chiarcos. 2009. ANNIS: A search tool for multi-layer annotated corpora. In *Corpus Linguistics 2009*.

Jan Štěpánek and Petr Pajas. 2010. Querying diverse treebanks in a uniform way. In *Proc LREC 2010*, pages 1828–1835.

Interactive Visualizations of Corpus Data in Sketch Engine

Lucia Kocincová[†]
xkocinc@fi.muni.cz

Vít Baisa[‡†] and **Miloš Jakubíček**[‡†] and **Vojtěch Kovář**[‡†]
name.surname@sketchengine.co.uk

[†]NLP Centre, Faculty of Informatics
Masaryk University, Brno, Czech Republic
[‡]Lexical Computing, Brighton, United Kingdom

Abstract

Automatic analysis of large text corpora produces large amounts of figures as result of various functions. These provide empirical evidence for a research hypothesis or serve in numerous practical applications of natural language processing. Usually, the results are presented in the form of tables containing raw data to be interpreted by domain experts. This paper describes an ongoing work on new visualizations and user interface enhancements in Sketch Engine corpus management system which aim at easing the interpretation of the data for both novice users and language professionals.

1 Introduction

Analyses of textual data deal with the issue of choosing a suitable representation for its results. While this factor is often neglected and most attention is being paid to the performance of the analytic functions (where the problem might be simply seen as continuation of Aristotle's *form vs. matter* debate, with matter being absolutely predominant in science), there is no doubt that the representation heavily influences how data are perceived and can significantly help or harm correct understanding of the results (Meirelles, 2013).

This becomes even more appealing where the underlying analytic sample does not posses uniform distribution—like language which usually follows Zipf's distribution (Zipf, 1949). Not only "ordinary" language users but sometimes even language experts tend to underestimate the impact of such heavily skewed distributions like the Zipfian one, henceforth drawing invalid conclusions from the analyses they carry out.

In this paper we describe an ongoing work on implementing new visualization options for language data analysis within Sketch Engine corpus management system (Kilgarriff et al., 2014). Sketch Engine has a large variety of users ranging from language learners, students and researchers in linguistics, lexicographers, translators and terminologists or data scientists in diverse domains.

The presented enhancements to the user interface are implemented with the hope that they will not only provide a more graphically appealing and easier way how to understand the data for novice users, but also speed up the daily work carried out by language experts (e.g. lexicographers).

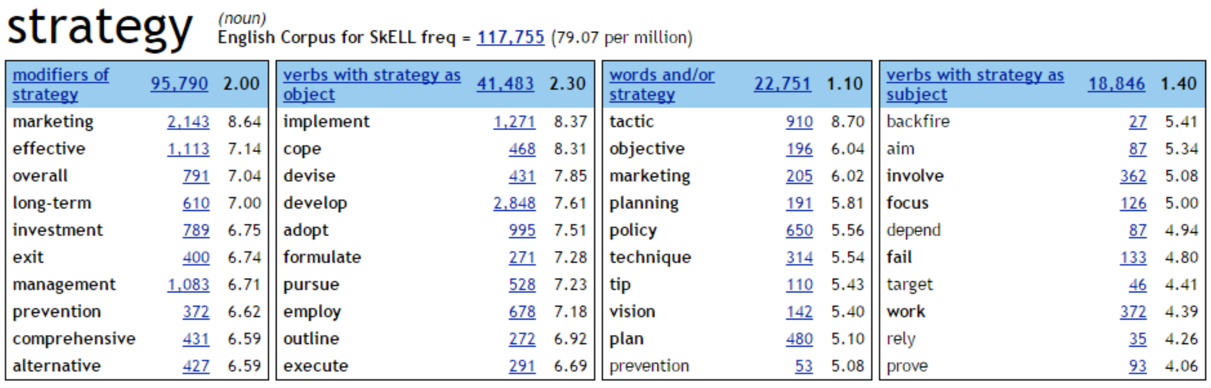
2 Sketch Engine

Sketch Engine is a leading corpus management system useful for discovering how words behave in different contexts. It has a wide range of analytic functions dealing with billion-word corpora (see e.g. (Pomikálek et al., 2012; Jakubíček et al., 2013)). In this paper we focus on the visualization of two core functions that leverage the principles of distributional semantics—word sketches and a distributional thesaurus.

2.1 Word Sketches

A word sketch is a one-page summary of a word's collocational behavior according to particular grammar relations. It is usually computed by evaluating a large number of corpus queries (Jakubíček et al., 2010) performing shallow parsing or by using some existing parser to do this task so as to retrieve a large number of collocation candidates which are then sorted using a lexicographic association score (see (Rychlý, 2008; Evert, 2005)).

A word sketch is currently presented in a table


strategy (noun)
English Corpus for SkELL freq = 117,755 (79.07 per million)

modifiers of strategy	95,790	2.00	verbs with strategy as object	41,483	2.30	words and/or strategy	22,751	1.10	verbs with strategy as subject	18,846	1.40
marketing	2,143	8.64	implement	1,271	8.37	tactic	910	8.70	backfire	27	5.41
effective	1,113	7.14	cope	468	8.31	objective	196	6.04	aim	87	5.34
overall	791	7.04	devise	431	7.85	marketing	205	6.02	involve	362	5.08
long-term	610	7.00	develop	2,848	7.61	planning	191	5.81	focus	126	5.00
investment	789	6.75	adopt	995	7.51	policy	650	5.56	depend	87	4.94
exit	400	6.74	formulate	271	7.28	technique	314	5.54	fail	133	4.80
management	1,083	6.71	pursue	528	7.23	tip	110	5.43	target	46	4.41
prevention	372	6.62	employ	678	7.18	vision	142	5.40	work	372	4.39
comprehensive	431	6.59	outline	272	6.92	plan	480	5.10	rely	35	4.26
alternative	427	6.59	execute	291	6.69	prevention	53	5.08	prove	93	4.06

Figure 1: Word sketches for lemma *strategy*.

where each column contains one grammar relation with collocates sorted by their score (see Figure 1).

2.2 Thesaurus

On top of the word sketches, Sketch Engine computes a distributional thesaurus. Normally a distributional thesaurus tackles the issue of finding similar words by asking the following question: given a word, what are the words that occur in the same context? In Sketch Engine this question is answered by finding words that share the same collocates in the same grammar relations in word sketches.

A thesaurus is simply a list of words accompanied with their frequency and a similarity score.

3 Thesaurus Visualization

The thesaurus already provides a visualization of the words in the form of a word cloud (Figure 2), however the score is not represented in the thesaurus in any way, so the potential of the data is not used to its full extent.

The main objective of the presented visualization in Figure 6 is to display all thesaurus attributes (textual string, its frequency and similarity and score) of each lemma in a meaningful but also a clear way, which was achieved by mapping these two values to multiple attributes. The core of the visualization is a lemma, the respective words are placed around it according to the score value—the higher the score is, the closer a word is to the center.

The score values are normalized into the $[0, 1]$ range, therefore a comparison of two different word lists is possible. However, a user evaluates each word list independently, so a fixed score axis could lead into misunderstanding in cases where score value is relatively low. This fact was taken into account when designing the visualization, so the score axis is adaptive – its boundary values are always taken from currently selected data, therefore the user can always clearly indicate the most similar words of the current lemma.

Score in the visualization is also mapped to a colour of the circle behind the word which simplifies comparison of two close words. The user can modify the colour range of score by choosing another appropriate colour from control panel, so the visual output can be adjusted.

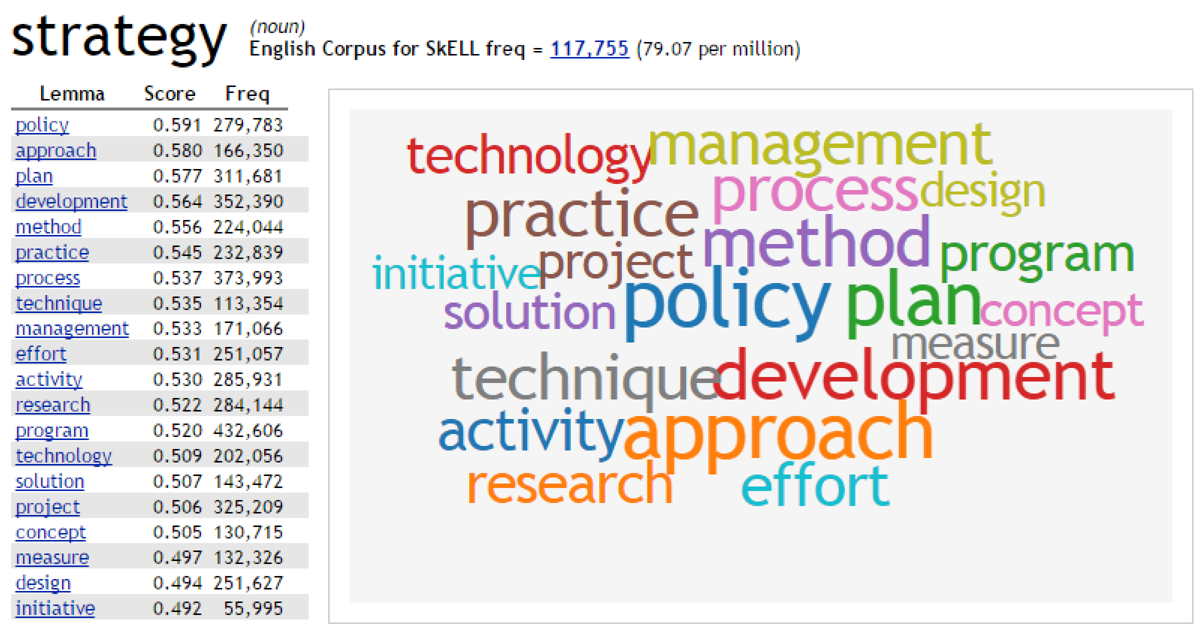
Frequency is mapped to the size of a circle and also to the font size of a word, which can be disabled to avoid confusion when comparing short and long words.

4 Word Sketch Visualization

Word sketches are technically very similar in terms of data types—thesaurus is a word list with score and frequency and word sketches are multiple word lists with the same attributes. Therefore, the principles in graphical representations may remain the same—assuming also that a consistent visualization across different system function makes its perception easier.

Score and frequency values are mapped to the same graphical elements as in thesaurus except the colour of circles because in word sketches, distinct colours are used for different grammar relations – as can be perceived from Figure 3. Therefore a score on different radius around the center is mapped to circle transparency so the words with the highest scores pop out from the center of the picture.

Example of word sketch visualization can be seen in Figure 4.


strategy (noun)
English Corpus for SkELL freq = 117,755 (79.07 per million)

Lemma	Score	Freq
policy	0.591	279,783
approach	0.580	166,350
plan	0.577	311,681
development	0.564	352,390
method	0.556	224,044
practice	0.545	232,839
process	0.537	373,993
technique	0.535	113,354
management	0.533	171,066
effort	0.531	251,057
activity	0.530	285,931
research	0.522	284,144
program	0.520	432,606
technology	0.509	202,056
solution	0.507	143,472
project	0.506	325,209
concept	0.505	130,715
measure	0.497	132,326
design	0.494	251,627
initiative	0.492	55,995

Figure 2: Thesaurus for lemma *strategy*.

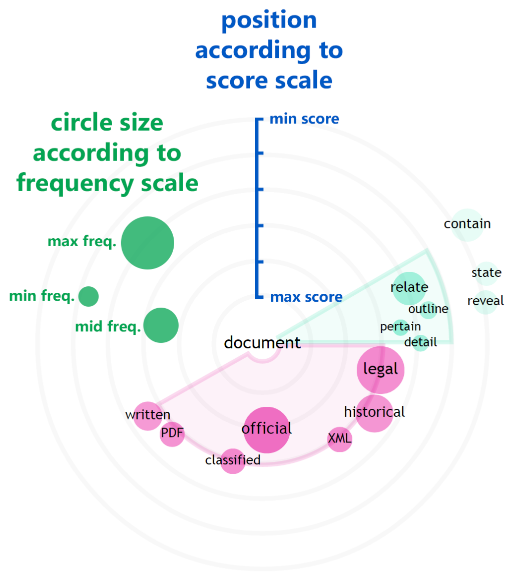


Figure 3: Mapping of graphical elements.

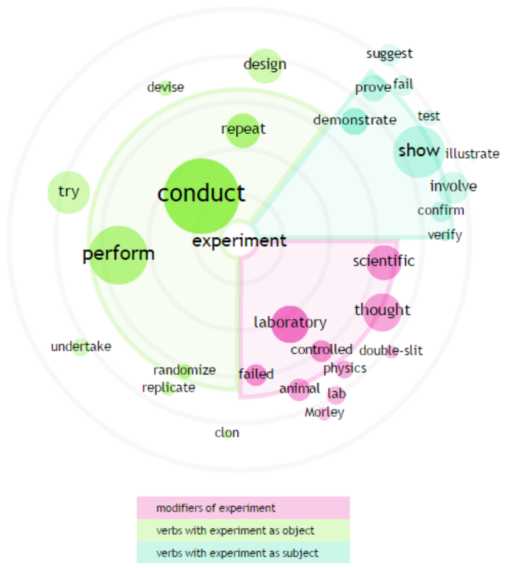


Figure 4: Word sketches of *experiment*.

5 Interactivity of Visualizations

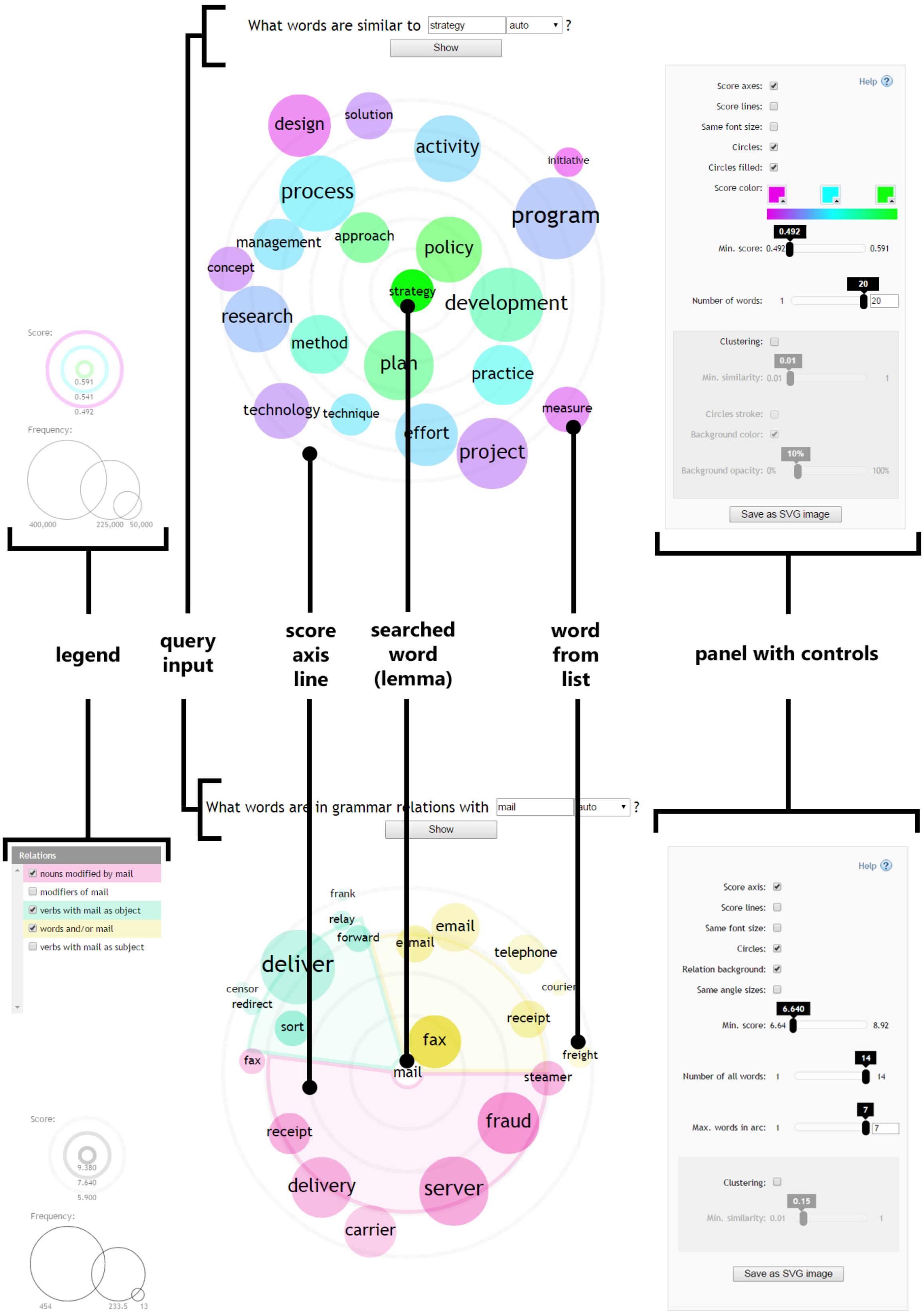
Each visualization is interactive thus the exploration of relationships among words is easier. All interaction controls are grouped in a panel located in the right side of the page. The panel includes as many options as possible, however the number of options will be reduced after a user testing.

The exact values of a word's frequency and score are not left out entirely, they can be retrieved on demand by hovering over a particular word. For an approximate evaluation of values which is mostly used by users, boundaries with labels are located in a legend. It is automatically updated when data changes so the user is always aware of the applied scale.

The interfaces and their parts are described in Figure 5.

6 Implementation

The visualizations introduced in this paper were implemented using JavaScript, jQuery and D3 library. D3 helps to focus on the graphical output



Score axes: ✓
Score lines: ☐
Same font size: ☐
Circles: ✓
Circles filled: ✓
Min. score: 0.492 (0.492 – 0.591)
Number of words: 20 (1 – 20)
Clustering: ☐
Min. similarity: 0.01 (0.01 – 1)
Circles stroke: ☐
Background color: ✓
Background opacity: 10% (0% – 100%)

✓ nouns modified by mail
☐ modifiers of mail
✓ verbs with mail as object
✓ words and/or mail
☐ verbs with mail as subject

Score axis: ✓
Score lines: ☐
Same font size: ☐
Circles: ✓
Relation background: ✓
Same angle sizes: ☐
Min. score: 6.640 (6.64 – 8.92)
Number of all words: 14 (1 – 14)
Max. words in arc: 7 (1 – 7)
Clustering: ☐
Min. similarity: 0.15 (0.01 – 1)

Figure 5: Visualization interface of thesaurus (top) and word sketches (bottom).

and its transformations and allows more effective development of interactive visualizations. (Bostock et al., 2011)

The input for scripts generating the visual output is a JSON object retrieved from Sketch Engine. Data boundaries and other properties are calculated to set up scales correctly.

An assignment of coordinates for each word is made afterwards according to score values. The new position has to satisfy two conditions—the word's bounding box cannot overlap with other bounding boxes in the area and also the circles cannot overlap. If these conditions are not met all scales are contracted and the positions are recalculated. If it is not possible to meet the given restrictions, for example too many words are being requested for output, the limitations are dropped and positions are calculated without any restrictions and it is upon the user to filter the output.

This behavior ensures that a visualization is always rendered and the user's requirements for data exploration are not limited.

The exact positions of words in a given score radius are currently random, but mapping of another attribute is possible in the future.

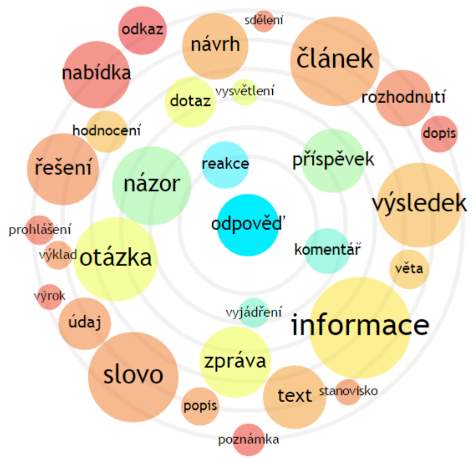
In the visualization of thesaurus the whole area around the center is covered with words. In the word sketches the whole area is divided into grammar relations, each relation is assigned an arc whose area is calculated as the sum of frequencies of words that belong to the given relation. The length of an arc represents the average score of displayed words.

The presented visualizations don't evaluate or modify the words as strings—the algorithms work with them as elements—they are therefore language-independent and can be used with any corpora available in Sketch Engine as can be seen in Figure 6.

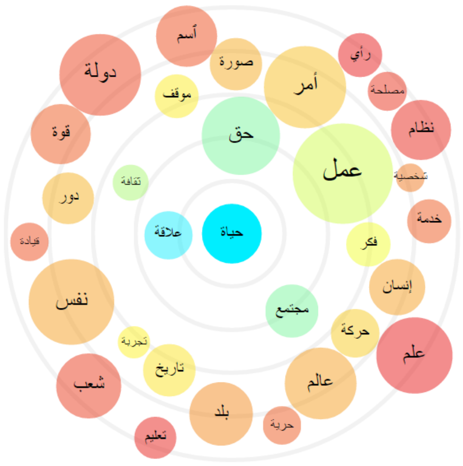
Graphics generated from the system can be also downloaded for further use.

7 Conclusions and Future Work

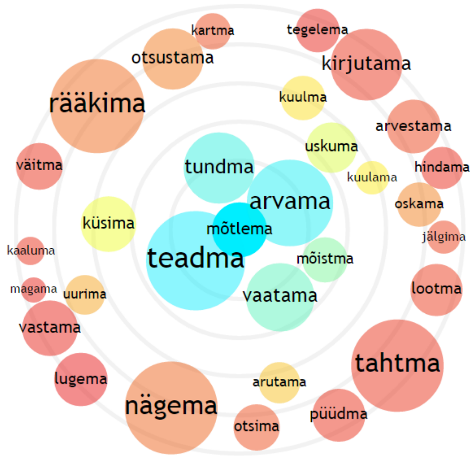
In this paper we have presented a number of visualization enhancements that are soon to appear in Sketch Engine (including its open source variant NoSketch Engine). The main added value of these visualizations is that users can immediately see differences between data values which can lead to faster interpretation of results and faster decisions. We are confident that further development of such



(a) Czech corpus



(b) Arabic corpus



(c) Estonian corpus

Figure 6: Thesaurus generated from different corpora.

enhancements is necessary to facilitate better understanding of the underlying corpus data especially in the context of ("big data").

Evaluation testing with differently experienced users of Sketch Engine is currently in progress and so far shows that the visualizations are mostly valuable for new and intermediate users. According to the feedback from users we also plan A/B testing to verify new improved versions of the presented visualizations. In the future further features of Sketch Engine will be subject to visualization enhancements as well (e. g. corpus metadata overview).

Acknowledgments

This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2010013. The research leading to these results has received funding from the Norwegian Financial Mechanism 2009–2014 and the Ministry of Education, Youth and Sports under ProjectContract no. MSMT-28477/2014 within the HaBiT Project 7F14047.

The access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005) is appreciated.

References

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 Data-Driven Documents. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2301–2309.

Stefan Evert. 2005. *The statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart.

Miloš Jakubíček, Adam Kilgarriff, Diana McCarthy, and Pavel Rychlý. 2010. Fast Syntactic Searching in Very Large Corpora for Many Languages. *PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 741–747.

Miloš Jakubíček, Adam Kilgarriff, Vojtěch Kovář, Pavel Rychlý, and Vít Suchomel. 2013. The TenTen Corpus Family. *International Conference on Corpus Linguistics, Lancaster*.

Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. The Sketch Engine: Ten Years On. *Lexicography*, 1:7–36.

Isabel Meirelles. 2013. *Design for Information: An Introduction to the Histories, Theories, and Best Practices Behind Effective Information Visualizations*. Rockport publishers.

Jan Pomikálek, Pavel Rychlý, and Miloš Jakubíček. 2012. Building a 70 Billion Word Corpus of English from ClueWeb. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, pages 502–506.

Pavel Rychlý. 2008. A Lexicographer-Friendly Association Score. *Proceedings of Recent Advances in Slavonic Natural Language Processing, RASLAN*, pages 6–9.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley Press.

Visualisation in Speech Corpora: Maps and Waves in the Glossa System

Michał Kosek, Anders Nøklestad, Joel Priestley, Kristin Hagen and Janne Bondi Johannessen
The Text Laboratory, Department of Linguistics and Scandinavian Studies
University of Oslo
Oslo, Norway
{michalkk,noklesta,joeljp,kristiha,jannebj}@iln.uio.no

Abstract

We present the Glossa web-based system for corpus search and results handling, focussing on two modes of visualisation implemented in the system. First, we describe the use of maps to show the geographical distribution of search results and its utility for exploring dialectal variation and discovering new isoglosses. Secondly, we present a functionality for speech visualisation, yielding dynamically generated representations of spectrograms, pitch and formants. The analyses are accompanied by the ability to replay selected parts of the waveform, as well as export and compare maximum, minimum and average values of the parameters for different selections. Among other things, this can be used to explore in more detail the set of spoken variants revealed by the geographical map view.

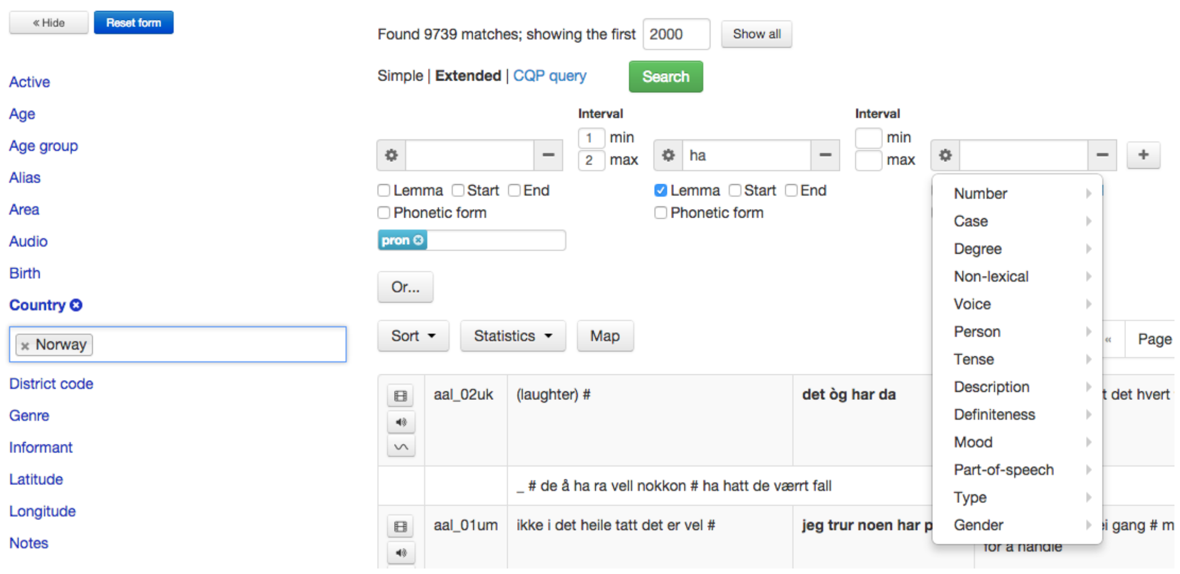
1 Introduction

The current availability of large corpora presents a new challenge for corpus research: how to process hundreds of millions or even billions of tokens, extract relevant information and transform it into a shape that can be used to explore specific hypotheses about language. In addition, the emergence of extensive collections of audio- and video-recorded speech, accompanied by transcriptions as well as metadata about the speakers such as their age, sex and geographical location, presents us with the challenge of how to make these additional sources of linguistic data readily available to language researchers. To help with these tasks, the Glossa corpus search system has been developed. The present article discusses the visualisation possibilities that Glossa provides for speech corpora, which are particularly useful for research within phonetics, phonology and dialectology.

The most important role of a speech corpus is to give access to data that are otherwise difficult to obtain. Without a corpus, researchers either need to trawl through a large amount of preexisting speech recordings, listening for words or patterns they are interested in, or elicit such patterns from speakers. The former option is very time-consuming, and the latter requires setting up an experiment, which may introduce a bias, since the experimental setup would be designed with specific research questions in mind. Speech corpora do not necessarily solve the observer's paradox (Labov, 1972). They do, however, provide speech data that are not affected by a specific research question. Furthermore, in some cases participants are not even aware that their speech will inform linguistic research. BigBrother[1] is one such corpus.

However, mere access to speech data is not sufficient. With the large amount of material currently available for many languages, techniques for extracting and visualising potentially interesting patterns in the data have become more important than ever. It is both important to have a way to group a large number of search results in a meaningful way, and to have the possibility to "zoom in" and analyse detailed features of the speech. If we take dialect research as an example, the corpus user may be interested in the relation between pronunciation and the place where the speakers live. The first of the presented features of Glossa groups the search results according to the phonetic transcription and geographical location and visualises them on a map. The user may then want to perform a more detailed analysis of pronunciation samples from some specific places shown on the map. Another feature of Glossa may then be used: the sound visualisation, which gives quick access to the spectrogram, waveform, pitch and formant

[1]`http://www.tekstlab.uio.no/nota/bigbrother/english.html`



aal_02uk	(laughter) #	det òg har da
	_ # de å ha ra vell nokkon # ha hatt de vært fall	
aal_01um	ikke i det heile tatt det er vel #	jeg trur noen har p

Figure 1: Advanced search in Glossa. We have searched in Norwegian dialects for a pronoun and the lemma *ha* separated by 1 or 2 tokens, and are in the process of selecting grammatical features for a third search term.

plots of the speech.

The article is structured as follows. Section 2 introduces general information about Glossa, and in particular, features related to speech corpora. The focus of the rest of the article is on visualisation possibilities of speech corpora in Glossa: geographical visualisation is presented in section 3, and speech visualisation in section 4. Section 5 describes technical details related to the implementation of these features. Then, section 6 discusses research applications that can benefit from these types of visualisation. Finally, section 7 discusses possible improvements of the features that may make research even more effective.

2 The Glossa Corpus Search System

2.1 Features of Glossa

Glossa is a web application that provides powerful methods for corpus search and result visualisation combined with a strong focus on user-friendliness. It allows a user to search monolingual and multilingual (parallel) text and speech corpora annotated with grammatical analyses or other types of token information. By selecting a set of metadata values (such as the author and publisher of a written text or the location and age of a speaker), the user can limit the search to a certain subcorpus.

There are three alternative search interfaces, ranging from maximum ease of use to maximally powerful queries:

a) a Google-like search box for simple token or phrase searches,

b) a set of text inputs, checkboxes and drop-down menus for more complex, grammatically specified searches (see Figure 1),

c) a search box for queries that are directly passed to the underlying search engine.

Results are presented as KWIC concordances, with the additional possibility to generate frequency distributions for tokens, lemmas or parts of speech. Glossa is easily installed on servers or laptops via Docker (see section 5). Alternatively, the source code can be freely downloaded from GitHub[2] under a very permissive open-source licence (MIT).

Out of the box, Glossa comes with support for corpora encoded with the IMS Open Corpus Workbench (Christ, 1994; Evert and Hardie, 2011)[3], which supports up to 2.1 billion tokens per corpus. However, Glossa was built from the ground up to be easily extended with support for different search engines and corresponding search and result views, and there is already an optional module for searching corpora on remote servers

[2] `https://github.com/textlab/glossa`
[3] `http://cwb.sourceforge.net`

using the Federated Content Search protocol defined by the European CLARIN infrastructure[4].

Glossa provides a simple admin interface which allows a Corpus Workbench (CWB) corpus to be created by uploading a zip file containing CWB indexes and potentially also tab separated value files with metadata as well as audio and video files if applicable. Glossa itself does not provide functionality to create these input resources; however we are currently working on a corpus processing pipeline for creating corpora from XML or plain text files, including TEI[5] format for written corpora and ELAN[6] format for speech corpora.

It should be noted that Glossa is not the only web-based corpus search system available; some examples of powerful alternatives are CQPweb[7], Corpuscle[8], Korp[9], and SketchEngine[10]. What sets Glossa apart from these is a unique combination of characteristics:

- the functionality for audio analysis and display of geographical distribution described in this paper,
- support for parallel queries in multilingual corpora,
- a strong focus on ease of use for non-technical users,
- ease of installation (particularly through its Docker distribution),
- extensibility with respect to different search engines and database systems,
- it is freely available without charge.

2.2 Speech Corpora in Glossa

With speech corpora, search results can be linked to audio and video clips that are accompanied by an auto-cue display showing each transcribed utterance as it is spoken. The utterances may have several different transcriptions. For example, in the Nordic Dialect Corpus (Johannessen et al., 2009), they are transcribed into the standard orthography, and to a simplified phonetic transcription, which shows how the word was actually pronounced in a particular utterance. The phonetic search feature allows looking for utterances where a word is pronounced in a particular way.

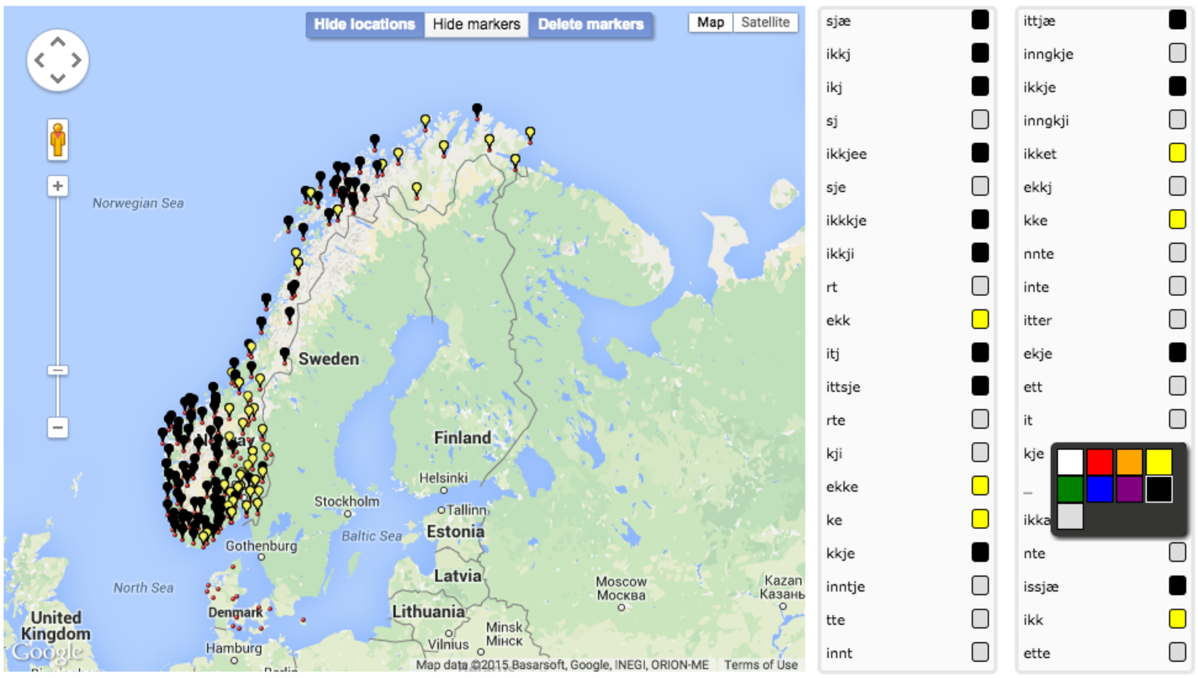
If the speakers in a corpus were recorded at different geographical locations (such as in a dialect corpus) and geographical coordinates are provided for these locations, search results can also be visualised as plots on a geographical map. Furthermore, if audio recordings are available, each search result can be analysed in an interface that implements the most important functionality found in desktop applications for sound analysis such as Praat (Boersma and Weenink, 2001). The rest of this paper will focus on the latter two functionalities: geographical maps and sound analysis.

3 Geographical Visualisation in Glossa

Corpus linguistic investigation commonly draws on analytic and communicative techniques taken from other fields. Dialectologists interested in regional variation have long turned to manually rendered maps to represent linguistic features. Perhaps the earliest example of such work is *Der Deutsche Sprachatlas*, carried out in the first half of the twentieth century and more recently digitised as a result of the *Digitaler Wenker-Atlas* (DiWA) project (Schmidt et al., 2001). However, as late as 2005, the lack of automation was still a concern (Labov et al., 2005, 41–42). Work has since been done to automatically render linguistic data in geographical maps, for example *Dynamic Syntactic Atlas of the Dutch dialects* (Barbiers and others, 2006, DynaSAND). The following section details one simple approach to achieving the type of digital linguistic topography heralded by Labov.

Geographical location is an essential metadata component of any speech corpus, and can be utilised in search visualisations. The Google Maps Embed API[11] has proven useful in extending the functionality of Glossa to this end. While originally incorporated into Glossa as a way of providing a metadata overview for corpus queries, it soon became evident that the spatial distribution of data reveals interesting patterns, particularly for corpora comprising multiple layers of transcription. The Norwegian component of the Nordic Dialect Corpus (Johannessen et al., 2009) is one such example; it is transcribed using a simplified phonetic

[4]http://clarin.eu
[5]http://www.tei-c.org/index.xml
[6]https://tla.mpi.nl/tools/tla-tools/elan
[7]https://cqpweb.lancs.ac.uk
[8]http://clarino.uib.no/korpuskel/page
[9]http://spraakbanken.gu.se/eng/korp-info
[10]http://www.sketchengine.co.uk
[11]https://developers.google.com/maps/web



sjæ	■	ittjæ	■
ikkj	■	inngkje	□
ikj	■	ikkje	■
sj	□	inngkji	□
ikkjee	■	ikket	🟨
sje	□	ekkj	□
ikkkje	■	kke	🟨
ikkji	■	nnte	□
rt	□	inte	□
ekk	🟨	itter	□
itj	■	ekje	■
ittsje	■	ett	□
rte	□	it	□
kji	□	kje	
ekke	🟨	_	
ke	🟨	ikka	
kkje	■	nte	□
inntje	□	issjæ	■
tte	□	ikk	🟨
innt	□	ette	□

Figure 2: Geographical visualisation in Glossa. We can see the geographical distribution of dialectal variations of *ikke* 'not' in Norway: yellow = velar plosive; black = fricative/affricate, non-nasal

system as its initial layer[12], with an automatically transliterated orthographic transcription providing a subsequent layer. This dual-tiered approach preserves the rich dialectal variation while ensuring searchability. Simple orthographic searches may yield dozens of distinct dialectal variants. Using the Embed API, Glossa takes advantage of this system of transcription by compiling the phonetic variants along with their corresponding geographical coordinates into a data structure and plotting the locations onto a Google map. Colour palettes are provided, giving the user the option of colour coding specified variants. The resulting clustering enables users to easily scan the distribution of linguistic features. For corpora of a sufficient size and geographical distribution, isoglosses are readily visible (see Figure 2). The size of the corpus, in terms of tokens per informant, will determine which linguistic features will be available for examination; less frequent features requiring more data. In the example, there is an average of about 4000 tokens per informant, with 564 informants spread amongst 163 locations.

[12]The transcription standard is described here: `http://www.tekstlab.uio.no/nota/scandiasyn/Transkripsjonsrettleiing%20for%20ScanDiaSyn.pdf` (in Norwegian). It is a coarse-grained transcription standard based on the Oslo Norwegian pronunciation of the alphabet (Papazian and Helleland, 2005).

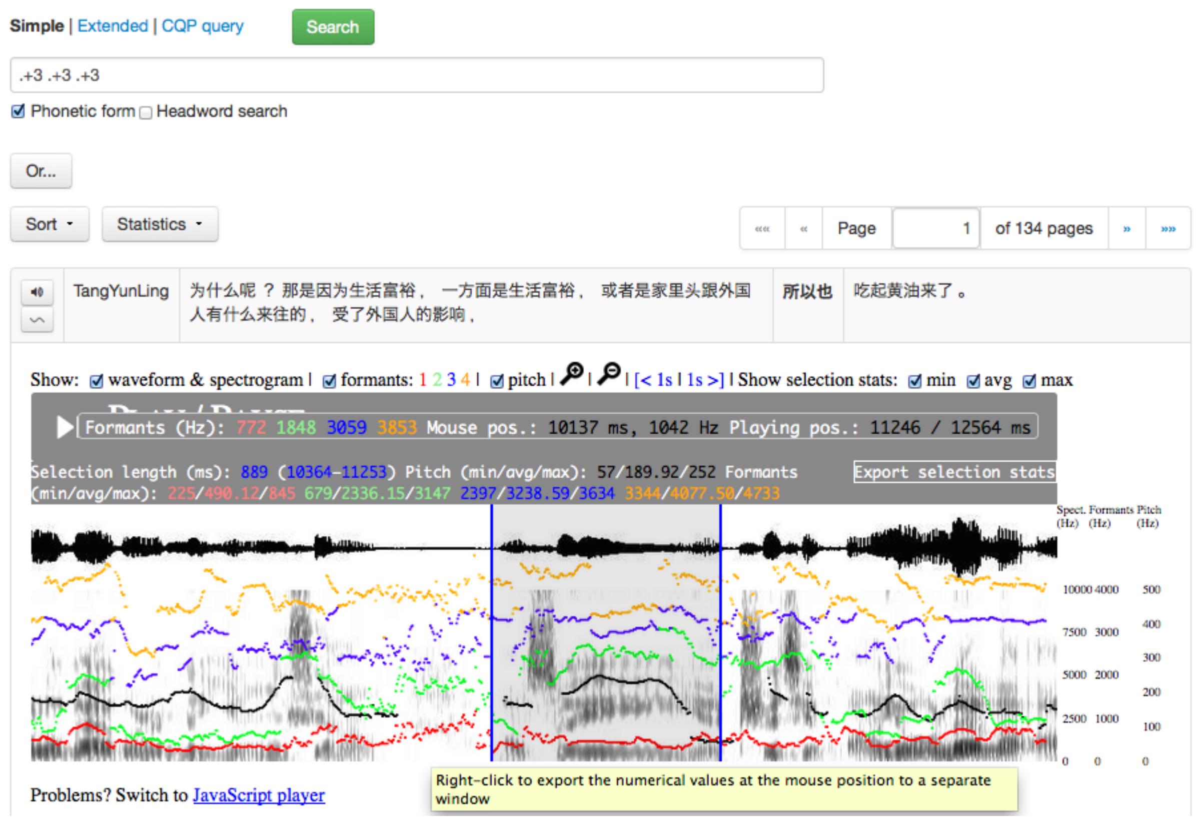
4 Speech Visualisation in Glossa

4.1 Background

Sound visualisation applications are an important part of the phonetician's toolkit; they provide information that does not depend on the subjective perception of the sound signal and can be presented and referred to in a written form. Among various sound analysis programs, Praat (Boersma and Weenink, 2001) is specifically designed to visualise and extract parameters of speech and therefore it is a standard tool within phonetics. Among its many features, the most significant ones include: visualisation of the waveform and the spectrogram, pitch and formant analysis.

Speech corpora not only enable easy access to a large amount of spoken utterances, but also allow the user to restrict the search according to a range of variables, based on corpus annotations and metadata. For instance, phonological annotations allow the user to find all words that share a particular pronunciation, part-of-speech tagging can be used to differentiate between some of the homonyms, and metadata may be used to specify sex and dialect of the speaker.

It would therefore be natural to use Praat to analyse search results from speech corpora. Unfortunately, it is often difficult or even impossible.


Simple | Extended | CQP query Search

.+3 .+3 .+3

☑ Phonetic form ☐ Headword search

Or...

Sort Statistics « « Page 1 of 134 pages » »»

TangYunLing 为什么呢 ？ 那是因为生活富裕， 一方面是生活富裕， 或者是家里头跟外国人有什么来往的， 受了外国人的影响， **所以也** 吃起黄油来了。

Show: ☑ waveform & spectrogram | ☑ formants: 1 2 3 4 | ☑ pitch | | | [< 1s | 1s >] | Show selection stats: ☑ min ☑ avg ☑ max

```
Formants (Hz): 772 1848 3059 3853 Mouse pos.: 10137 ms, 1042 Hz Playing pos.: 11246 / 12564 ms

Selection length (ms): 889 (10364-11253) Pitch (min/avg/max): 57/189.92/252 Formants
(min/avg/max): 225/490.12/845 679/2336.15/3147 2397/3238.59/3634 3344/4077.50/4733
```

Export selection stats

Right-click to export the numerical values at the mouse position to a separate window

Problems? Switch to JavaScript player

Figure 3: Sound visualisation in Glossa. We have searched a Mandarin Chinese corpus for utterances with three consecutive third tones. The pitch plot (black) shows that in the presented search result only the last syllable is pronounced with the actual low third tone. The figure also shows waveform, spectrogram and formant plots, and the user interface that provides functionalities described in the article.

Praat operates only on local files saved on the hard drive. Due to copyright restrictions, speech corpora do not commonly allow sound files to be directly downloaded. Most often the search results are only available via streaming and can only be listened to through a web browser.

If the download links are available, Praat might be sufficient for research that requires deep analysis of a relatively small amount of samples. There are, however, many situations in which the use of Praat is suboptimal. For instance, in the exploratory phase of the research, when one tries to formulate a hypothesis, it is often beneficial to conduct a lot of different searches according to different criteria and perform a quick analysis of the results. In such cases, downloading each result separately and repeatedly switching between separate applications is ineffective.

4.2 Features of the Current Visualisation System

The online speech visualisation in Glossa contains the most important features of Praat and solves the above-mentioned problems. The sound visualisation is accessible through an icon next to each result presenting KWIC concordances. An overview of the user interface is presented in Figure 3.

The following plots are available: the waveform in the upper part, the spectrogram in the lower part, and four formants and pitch overlaid over the spectrogram. The plots can be turned on or off individually according to user preference. Just like in Praat, the sound can be played and an animated vertical line shows the current playing position in real time.

The user can select a part of the spectrogram that may, for example, correspond to one sound or syllable, listen to that part, and zoom in to see the features of the sound in more detail. As the selection is made, the numerical values of the parameters are displayed: duration and statistics for the selected period, namely maximal, minimal and average value of the pitch and formants. The time and frequency of any point on the spectrogram can be accessed via a mouse hover-over. Right-clicking allows the numerical values to be exported to a separate window. Also the statistics of the selection can be exported.

The visualisation is fully integrated with the search functionality of Glossa. For instance, in a corpus with speaker metadata, part-of-speech annotation and phonetic transcription, the user may specify a particular word in its orthographic and/or phonetic transcription, its part of speech, and restrict the search to utterances from speakers who meet particular criteria.

5 Technical Details

Even though Praat has batch processing facilities, it does not provide any application programming interface (API) that would make integration into a larger system possible. Therefore we used the Snack Sound Toolkit[13] instead, which contains Tcl/Tk and Python bindings.

The visualisation is generated by a Python daemon using Snack, which communicates with the main Glossa daemon, written in Ruby on Rails, through a simple protocol. Such an architecture is motivated by several factors. Snack does not have Ruby bindings, so calling it directly from the main Glossa daemon is not possible. Moreover, having this functionality in a separate daemon allows Glossa to be more responsive: other functionalities of Glossa are not blocked during the generation of the spectrograms, nor is it necessary to create a separate process for every request. Additionally, initialisation of Snack and other related code requires several seconds. In our architecture this needs to be done only once, after starting the daemon, and does not affect the time of the generation of the spectrogram.

The speed of the spectrogram generation depends on the size of the segments[14]. The daemon can generate spectrograms for sound chunks of any length. Since Snack needs to load the whole file into memory in order to generate the spectrogram, and the generation of some of the plots (e.g. the formants) may be time-consuming for longer files, we decided to split sound files into one-minute chunks, generate the plots for each of them, and stitch them together in a way that makes the result practically indistinguishable from a plot generated directly from a larger segment. This solution makes visualisation of large sound files possible without using too many resources. That said, if the chunks are too long, the system becomes less usable for the researchers, as they need to spend time looking for words or sounds they are interested in. In our corpora, the segments are generally not longer than a few dozens of seconds. The plot for a typical, 10-second segment is generated on our server in less than 4 seconds, which allows high interactivity. If a corpus is divided into larger segments, the user will need to wait a bit longer, but our solution is still faster than downloading the file and opening it in Praat: the plot for a 38-second segment is generated in less than 9 seconds, and for a 94-second segment in less than 21 seconds.

The interactive audio player that allows any part of the displayed sound to be played was written in JavaScript and makes use of the API provided by SoundManager 2[15]. Our visualisation tool may use Adobe Flash to play sounds, or use a purely JavaScript-based alternative when Flash is not available.

In order to minimise installation problems and support reproducible research (Chamberlain and Schommer, 2014), Glossa is released as a Docker[16] image, which contains all the required dependencies. Such packaging guarantees that it will give exactly the same results on the same data, avoiding problems caused by different software versions. Moreover, although the system makes use of several programming languages and libraries, due to Docker packaging, it is cross-platform and installable with just a few clicks.

In order to render the maps, Glossa communicates with the Google Maps Embed API using a JSON object. The colour-coding widget is borrowed from the JQuery API, and shares the same JSON object. The reason for choosing Google's service was simply practicality. At the time of implementation, Google's API was deemed most well developed and its interface most familiar to potential users. However, most of the development involved was agnostic with regards to which service was chosen, meaning any future move to another service will require little adaptation. It is worth noting here that using an open-source map service, such as OpenStreetMap, would enable hosting the maps together with the Glossa in-

[13]`http://www.speech.kth.se/snack`

[14]A segment is a piece of transcription synchronised with the audio, with a defined start and end time.

[15]`http://www.schillmania.com/projects/soundmanager2`

[16]`http://www.docker.com`

stallation, thus enabling offline viewing.

6 Research Applications

As mentioned in subsection 4.1, the visualisation features of Glossa may be useful in the exploratory phase of the research. For example, one may start by searching for a word in the Nordic Dialect Corpus and see the geographical distribution of different pronunciation variants. The phonetic search feature, mentioned in subsection 2.2, may then be used to restrict the search to one of the variants presented by the geographical visualisation. The phonetic transcription used in the corpus is coarse-grained, which means that there may be subtler differences within a particular variant. The sound visualisation allows the user to find out whether there are differences within the chosen pronunciation variant. For instance, one may measure duration of the vowels or the voice onset time of the consonants. The spectrogram analysis may improve accuracy in differentiating relatively similar phones (such as alveolar tap and trill), compared to a situation where only audio is available. The tool is designed to work interactively, and one may easily repeat the procedure with different words and different features in order to produce a hypothesis.

The map visualisation has already had a significant impact on the Scandinavian linguistic research community. Researchers from the projects NorDiaSyn and NorDiaCorp have written more than sixty papers on various syntactic phenomena in the North Germanic languages, in which maps from the Nordic Dialect Corpus have played a crucial role. These papers have been published in a new online open access journal which requires its papers to use empirical data from, *inter alia*, the Nordic Dialect Corpus and maps generated from it: *Nordic Atlas of Language Structures Journal* (Johannessen and Vangsnes, 2014).

Visualisation in speech corpora is also useful for more reliable data gathering. For example, stress is a feature of spoken Mandarin Chinese that received relatively little attention. One may want to use a Mandarin speech corpus, such as MAID[17], to investigate the patterns of its occurrence. The problem, however, is that stress is not marked in the Chinese writing system, and *Hanyu Pinyin*, the official phonetic transcription system for Mandarin, only distinguishes unstressed syllables that completely lose their underlying tone. Therefore, without a sound recording it is impossible to distinguish unstressed syllables that still retain their tone from stressed syllables. Another feature of Mandarin, present in the Beijing dialect, is the retroflex suffixation. This suffix is often omitted in speech transcription, and therefore an actual recording is again more reliable than transcription alone. But even with a recording, researchers often have no other choice than to rely on their subjective evaluation of whether stress or a suffix is present in a particular syllable. The sound visualisation makes it possible to refer to objective features of the waveform, spectrogram and formants, such as F3 decrease in case of the retroflex suffixation (Lee, 2005), instead of researchers' subjective perception.

Those who work with specific groups of speakers, for example children or people with pronunciation difficulties, may take advantage of features of Glossa, even if corpora covering their target group are not available. For example, Norwegian children may tend to produce epenthetic vowels in word-initial consonant clusters. Researching this feature requires a control group of adult Norwegian speakers, and this is where data from the Norwegian Speech Corpus[18] or the Nordic Dialect Corpus[19] may be useful. The sound visualisation makes it possible to quickly find out whether adults produce such epenthetic vowels and to measure their duration.

One of the easiest features to analyse in the sound visualisation is the pitch contour, which may give valuable information, especially in the case of tonal languages. For example, one may analyse the 3rd tone sandhi in Mandarin Chinese: the patterns occurring when there are two or more subsequent syllables with the 3rd tone. When the syllables occur within a prosodic foot, all but the last one change to the 2nd tone (Shih, 1997). In other cases the change is not obligatory, but may occur. The pitch plot is a useful tool for verification of whether the tone sandhi actually occurs and analyse patterns of its occurrence. The visualisation is even more useful for investigating effects of tone coarticulation – while the tone sandhi is categorical, the coarticulation effect changes the pitch in different degrees, depending on the sit-

[17]Mandarin Audio Idiolect Dictionary, a dictionary and corpus of Beijing Mandarin: `http://www.hf.uio.no/iln/om/organisasjon/tekstlab/prosjekter/maid`

[18]`http://tekstlab.uio.no/nota/oslo/english`

[19]`http://tekstlab.uio.no/nota/scandiasyn`

uation (Zhang and Liu, 2011). Figure 3 shows how one may use Glossa to look for particular tone combinations, visualise and analyse the pitch contour. Average pitch values over specified periods of time can then be exported, which allows for the investigation of degrees of tone changes in natural speech, depending on the adjacent tonal context.

No formal evaluation of the presented visualisation features has been performed. However, the fact that many researchers use the corpora served by Glossa on a daily basis, and publish research papers based on them, is a sign that they have succeeded in satisfying the user groups. 16,142 individual searches were performed by more than 220 different users between 20. December 2014 and 19. April 2015 (i.e. 135 searches per day on average). The many papers in the *Nordic Atlas of Language Structures Journal* provide yet more evidence. The quantitative results from Google Scholar are also worth mentioning: 141 scholarly publications refer to "Nordic Dialect Corpus", which is just one of the many corpora using Glossa, and another 24 refer to its Norwegian name "Nordisk dialektkorpus". These are high numbers for a corpus that was only ready for use in 2011.

7 Future Work

The detection of isoglosses discussed still leaves the job of charting them. One interesting line of future development would be attempting to perform this task automatically. The application of k-means clustering and computing convex hulls (Wiedenbeck and La Touche, 2008) would be one such avenue.

The currently available sound visualisation is a multi-purpose tool that provides a wide range of acoustic data about the speech signal. It reduces the time required to visualise the features of the utterances that are of interest. However, corpora should not only give access to specific examples, but also provide useful statistics that allow generalisations to be drawn from the search results. In this case, the efficiency of research would be increased even more if the corpus search tool could directly provide statistics relating to the search results, for example average voice onset time, vowel length or formant values within a vowel. This would, however, require synchronisation of speech at the level of words and/or syllables.

In the speech corpora currently available in Glossa, the sound and the text are synchronised at the utterance level. There are, however, no technical problems with producing corpora that are synchronised at the word level. When such corpora become available, Glossa may be extended with an API that allows algorithms that find and/or measure specific phonetic details to be applied, such as the automatic measurement of voice onset time (Sonderegger and Keshet, 2012) or the detection of retroflex suffixation (Zhang et al., 2014).

8 Conclusion

This paper has discussed the visualisation possibilities of the Glossa corpus search system. The main focus was on the features available for speech corpora: geographical visualisation and speech visualisation. Geographical visualisation makes it possible to display pronunciation variants of the search results on a map and use colour-coding to cluster them into larger groups. The phonetic search feature allows specific pronunciation variants in the corpus to be found. Each search result in a speech corpus can be visualised in the built-in tool for audio analysis. The user may select its part and plot its parameters or export their values. These visualisations may be used to explore the data and formulate a research hypothesis, verify the existence of particular phonetic features, and easily analyse various parameters of speech.

Acknowledgements

We would like to thank Professor Christoph Harbsmeier for his suggestions and input on the visualisation of speech, and Pernille Hansen and Anders Vaa for their ideas on the use of the speech features now included in Glossa. Further, we are grateful to those who have provided speech content, and to the transcribers who have facilitated the searchability of our corpora. The two projects NorDiaSyn (financed by the Research Council of Norway) and NorDiaCorp (financed by Nordforsk) were responsible for many of the recordings and the transcriptions used in the present speech corpora.

This work was partly supported by the Research Council of Norway through its Centres of Excellence funding scheme, project number 223265, and partly through the Research Council's infrastructure project CLARINO.

References

Sjef Barbiers et al. 2006. *Dynamic Syntactic Atlas of the Dutch dialects (DynaSAND)*. Meertens Institute, Amsterdam. `http://www.meertens.knaw.nl/sand`.

Paul Boersma and David Weenink. 2001. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–345.

Ryan Chamberlain and Jennifer Schommer. 2014. Using Docker to support reproducible research. Technical report, Invenshure, LLC. `http://dx.doi.org/10.6084/m9.figshare.1101910`.

Oliver Christ. 1994. A modular and flexible architecture for an integrated corpus query system. In *Proceedings of the 3rd International Conference on Computational Lexicography (COMPLEX)*, pages 22–32, Budapest.

Stefan Evert and Andrew Hardie. 2011. Twenty-first century Corpus Workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 conference*, Birmingham. University of Birmingham.

Janne Bondi Johannessen and Øystein Alexander Vangsnes. 2014. *Nordic Atlas of Language Structures Journal*. Department of Linguistics and Scandinavian Studies, University of Oslo. `http://www.tekstlab.uio.no/nals/`.

Janne Bondi Johannessen, Joel Priestley, Kristin Hagen, Tor Anders Åfarli, and Øystein Alexander Vangsnes. 2009. The Nordic Dialect Corpus – An advanced research tool. In *Proceedings of the 17th Nordic Conference of Computational Linguistics NODALIDA 2009. NEALT proceedings series*, volume 4.

William Labov, Sharon Ash, and Charles Boberg. 2005. *The atlas of North American English: Phonetics, phonology and sound change*. Walter de Gruyter.

William Labov. 1972. *Sociolinguistic patterns*. Number 4 in Conduct and Communication. University of Pennsylvania Press.

Wai-Sum Lee. 2005. A phonetic study of the “er-hua” rimes in Beijing Mandarin. In *Ninth European Conference on Speech Communication and Technology*.

Eric Papazian and Botolv Helleland. 2005. *Norsk talemål*. Hyskoleforlaget, Kristiansand.

Jürgen Erich Schmidt, Joachim Herrgen, Tanja Giessler, Alfred Lameli, Alexandra Lenz, Karl-Heinz Müller, Wolfgang Näser, Jost Nickel, Roland Kehrein, Christoph Purschke, et al. 2001. *Digitaler Wenker-Atlas*. Forschungszentrum Deutscher Sprachatlas, Marburg. `http://www.diwa.info`.

Chilin Shih. 1997. Mandarin third tone sandhi and prosodic structure. *Studies in Chinese Phonology*, 20:81–123.

Morgan Sonderegger and Joseph Keshet. 2012. Automatic measurement of voice onset time using discriminative structured prediction. *The Journal of the Acoustical Society of America*, 132(6):3965–3979.

Bryce Wiedenbeck and Kit La Touche. 2008. Drawing isoglosses algorithmically. In *Class of 2008 Senior Conference on Computational Geometry*, page 22.

Jie Zhang and Jiang Liu. 2011. Tone sandhi and tonal coarticulation in Tianjin Chinese. *Phonetica*, 68:161–191.

Long Zhang, Haifeng Li, Lin Ma, and Jianhua Wang. 2014. Automatic detection and evaluation of Erhua in the Putonghua proficiency test. *Chinese Journal of Acoustics*, 1:83–96.

The ParaViz Tool: Exploring Cross-linguistic Differences in Functional Domains Based on a Parallel corpus

Ruprecht von Waldenfels
Institute of Polish,
Polish Academy of Sciences, Cracow
ruprecht.waldenfels@gmail.com

Abstract

ParaViz is a modular corpus query and analysis tool in development for use with a word-aligned, linguistically annotated multilingual parallel corpus. Representing an addition to classic query-based corpus tools, it allows to assess the cross-linguistic variation in the functional domain of items or structures that are defined as cognate or otherwise equivalent by the user. ParaViz provides the user with two perspectives on such data: on the one hand, a close-up perspective with word-aligned corpus examples that are classified and color-coded according to the user's criteria; on the other hand, a bird's view perspective with word lists and Neighbor-Net visualizations that offer an overview of the aggregated differences in use. Together they enable researchers to quickly find and explore convergent and divergent functional patterns of equivalent formants in different languages.

1 Introduction

ParaViz is a modular corpus query and analysis tool in development for use with a word-aligned, linguistically annotated multilingual corpus. It is deployed with ParaSol, a small multilingual parallel corpus primarily geared towards linguistic contrastive and typological research of Slavic (Waldenfels 2011), but may in general be used with any massively parallel word-aligned corpus such as Opus (Tiedemann, 2012) or InterCorp (Čermák and Rosen, 2012). ParaViz functions as a stand-alone component at the moment of writing and is planned to be implemented as a web application.

ParaViz adds a new type of functionality to parallel corpus querying going beyond what is described in Volk et al. (2014). It supplements and builds on ParaVoz (Meyer, Waldenfels, Zeman 2014), a corpus query interface which allows querying parallel corpora through a traditional web interface on the basis of complex queries involving token sequences across aligned languages, including negative queries on aligned segments.

Section 2 of this paper gives an introduction to functional comparison using parallel texts as implemented in ParaViz. In section 3, I describe the implementation, and conclude in section 4.

2 Functional Comparison Based on Parallel Texts

The function of a linguistic item as understood here includes its semantic, pragmatic, or other characteristics. As a rule, functional characteristics of linguistic elements are harder to describe than their formal characteristics and involve complex analysis of corpus examples. This makes the comparison of such functions particularily difficult, since it presupposes a consistent, comparable analysis of these functions across many languages, a task that quickly becomes very complex with a growing number of languages.

To see this, consider a comparison of the German, Dutch, and English perfect. In all three languages, it is quite straightforward to describe cognate perfect constructions that consist of an auxiliary ('have' or 'be') and a past participle. However, in order to compare the functional profiles of such constructions, we first need to describe the functions of each construction, taking care to do this in a consistent, comparable manner that is indeed relevant for the comparison and does not miss important contrasts. This is a very time-consuming and difficult task for even a medium number of languages.

The fact that aligned parallel corpora involve translationally equivalent texts in many languages can be harnessed to quickly attain insights on functional differences and similarities based on formal definitions alone (see the papers in Cysouw and Wälchli (2007) and Dahl (2014) for related approaches). The basic notion is straightforward: if two items in two languages are often used as translations of each other, we assume that their functional potential is similar. This notion can be used to do quite complex comparisons of functional domains. In application to the above example, the fact that these perfect constructions differ in their function quickly emerges from simply observing that their distribution is very different in the parallel corpus.

For a different, rather lexical example, let us assume users want to compare the functional domain of color terms in different languages. In order to do that, users define which words represent the lexical categories red, blue and yellow across many languages, and the system then compares the use of these words (and thus, the lexical categories) across languages in translationally equivalent expressions. For example, such a comparison would show that German *blau*, English *blue*, French *bleu* are often used in the same segments, but that the distribution of Russian *sinij* stands apart, since this item denotes a dark hue of blue. Moreover, the German representative of the lexical concept 'blue' is used in contexts that it isn't used in English and French since it also refers to a state of drunkenness (*er ist blau* 'he is drunk'), while in English, *blue* also denotes a melancholic state of mind (as in *I'm feeling blue*). If these uses are attested in the parallel corpus, the differences in the denotation of hues as well as in non-literal uses are readily apparent in the distribution of these terms in translationally equivalent segments across the languages in question.

In this way, the functions of variables of different types ranging from grammatical categories such as tense, aspect, or case to lexical categories such as words for the color red or derivational suffixes can be easily compared. Further applications and a more detailed description of the approach are found in von Waldenfels (2014).

3 The ParaViz system

ParaViz is meant to simplify the type of comparison outlined in section 2 by offering a standardized way to easily conduct such functional comparisons for a wide range of variables. This is done by offering the user a mechanism to define variables in a formal way and outputing the results of a classification of the corpus data based on these definitions. This section describes the standalone application as it is functional at the moment of writing; in the future, this system is planned to be implemented as a web service.

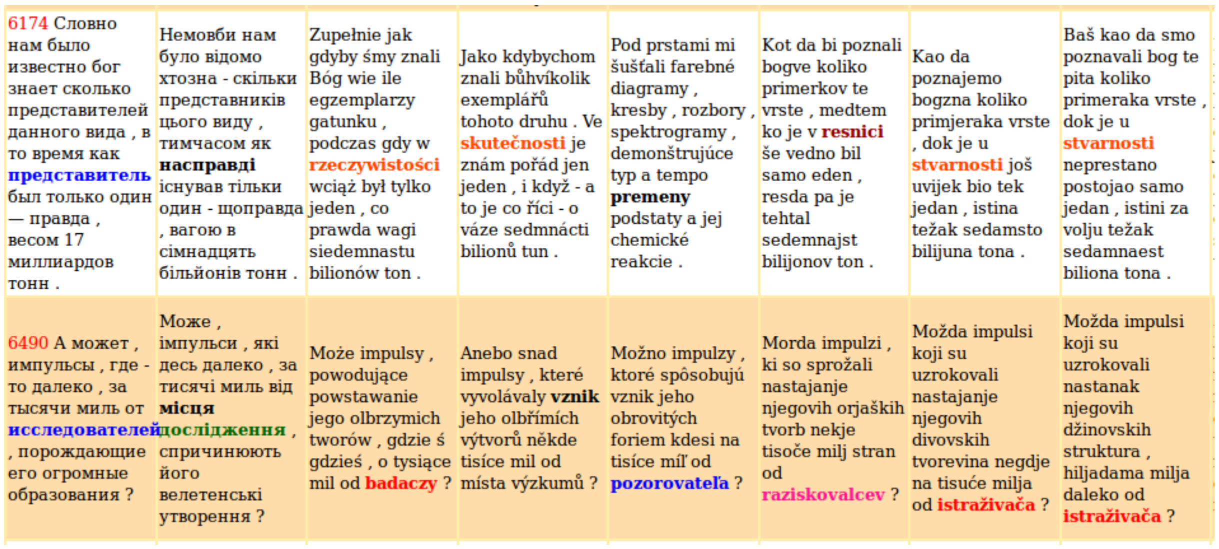
ParaViz is used with ParaSol, a small multilingual parallel corpus primarily geared towards linguistic contrastive and typological research[1]. ParaSol focusses on Slavic, but also includes Romance, Germanic, Finno-Ugric, Greek, Armenian and other languages. The word forms in most languages are lemmatized and POS-tagged; a subset of the corpus is word-aligned using UPLUG (Tiedemann, 2003).

3.1 Operationalization of Variables

As a first step in the process, users define the sets of elements which they want to compare across languages. This is done by the operationalization of variables in parameter files in XML format. In such a paramter file, variables are defined as constraints over word-aligned word forms and their annotation. At the moment, the parameter files allow the definition of such variables as regular expressions over tokens, their lemmas and POS tags, as well as over tokens, lemmas and POS tags directly adjacent. The following example defines the suffix classes OST and STVO, both denoting abstract nouns, in two Slavic languages:

```
<parameter id="NounSuffixes">
<type id="O" name="OST">
 <criteria><lng>ru</lng>
  <regexp level="lem">ость$</regexp>
  <regexp level="tag">^N.*</regexp>
 </criteria>
...
 <criteria><lng>sl</lng>
  <regexp level="lem">ost$</regexp>
  <regexp level="tag">^S.*</regexp>
 </criteria>
</type>
<type id="S" name="STVO">
 <criteria><lng>ru</lng>
  <regexp level="lem">ство$</regexp>
  <regexp level="tag">^N.*</regexp>
 </criteria>
...
 <criteria><lng>pl</lng>
  <regexp level="lem">[cs]two$</regexp>
  <regexp level="tag">^subst.*</regexp>
```

[1]http://www.parasolcorpus.org



6174 Словно нам было известно бог знает сколько представителей данного вида , в то время как **представитель** был только один — правда , весом 17 миллиардов тонн .	Немовби нам було відомо хтозна - скільки представників цього виду , тимчасом як **насправді** існував тільки один - щоправда , вагою в сімнадцять більйонів тонн .	Zupełnie jak gdyby śmy znali Bóg wie ile egzemplarzy gatunku , podczas gdy w **rzeczywistości** wciąż był tylko jeden , co prawda wagi siedemnastu bilionów ton .	Jako kdybychom znali bůhvíkolik exemplářů tohoto druhu . Ve **skutečnosti** je znám pořád jen jeden , i když - a to je co říci - o váze sedmnácti bilionů tun .	Pod prstami mi šušťali farebné diagramy , kresby , rozbory , spektrogramy , demonštrujúce typ a tempo **premeny** podstaty a jej chemické reakcie .	Kot da bi poznali bogve koliko primerkov te vrste , medtem ko je v **resnici** še vedno bil samo eden , resda pa je tehtal sedemnajst bilijonov ton .	Kao da poznajemo bogzna koliko primjeraka vrste , dok je u **stvarnosti** još uvijek bio tek jedan , istina težak sedamsto bilijuna tona .	Baš kao da smo poznavali bog te pita koliko primeraka vrste , dok je u **stvarnosti** neprestano postojao samo jedan , istini za volju težak sedamnaest biliona tona .
6490 А может , импульсы , где - то далеко , за тысячи миль от **исследователей** , порождающие его огромные образования ?	Може , імпульси , які десь далеко , за тисячі миль від **місця** **дослідження** , спричинюють його велетенські утворення ?	Może impulsy , powodujące powstawanie jego olbrzymich tworów , gdzie ś gdzieś , o tysiące mil od **badaczy** ?	Anebo snad impulsy , které vyvolávaly **vznik** jeho olbřímích výtvorů někde tisíce mil od místa výzkumů ?	Možno impulzy , ktoré spôsobujú vznik jeho obrovitých foriem kdesi na tisíce míľ od **pozorovateľa** ?	Morda impulzi , ki so sprožali nastajanje njegovih orjaških tvorb nekje tisoče milj stran od **raziskovalcev** ?	Možda impulsi koji su uzrokovali nastajanje njegovih divovskih tvorevina negdje na tisuće milja od **istraživača** ?	Možda impulsi koji su uzrokovali nastanak njegovih džinovskih struktura , hiljadama milja daleko od **istraživača** ?

Figure 1: A word-aligned corpus sample with color coding according to user-supplied parameter file.

```
</criteria>
</type>
```

The user then defines a filter condition for one of the languages that is taken as the primary language. For example, for the comparison of nominal suffixes above, the user may choose to classify only nouns. In other cases, the user may want to restrict the classification to some list of lemmata; for example, the user may be interested in a particular lexical or grammatical domain. The primary language will usually be the language of the original, but in general, any language may be chosen.

The tokens in the primary language that satisfy the filter condition as well as well their word-aligned equivalents in the other languages are classified in the next stop. This classification assigns a type to each token in question based on the criteria defined in the parameter file.

The system then does a corpus search and classification of the corpus data, which is offered to the user in two ways; first, as random samples of the corpus hits in context; second, in an aggregate form.

3.2 Qualitative Perspective: Color-coded Corpus Samples

As a first result, the user is given random samples of corpus data conforming to the definitions. This enables the user to review and refine the operationalization of his or her parameters. The word-aligned forms in these corpus results are color-coded to reflect the types previously defined in the parameter file; an example output is given in figure 1. This perspective affords a qualitative assessment and allows the user to explore the data.

This part of the output is based on ParaVoz, a modular corpus query interface for CorpusWorkbench[2] (CWB; see Evert & Hardie (2011)) published as open source (Meyer et al. 2014). It is designed as an easy to use, easy to install and easy to maintain flexible corpus interface for a parallel corpus hosted by CWB. ParaVoz (and CWB) uses CQP, a query language also used with a number of other corpus engines such as the NoSketchEngine (Rychlý, 2007) or Poliqarp[3]. ParaVoz does not use the CWB output directly, but, having configured CWB to SGML mode, reformats its output into a convenience XML format using regular expressions.

The output module of ParaVoz then uses XSLT to transform the XML result document. Using XML and XSLT at this stage allows rapid adaption to diverse types of corpus data. In this case, word alignment visualization is realized by linking ids that are encoded as token annotations. These annotions are compared, and if a token in one of the target languages is aligned to the target tokens in the source language, it is shown in bold. Using the same script, each target form is classified according to the user-defined definitions and color-coded accordingly.

[2]http://cwb.sourceforge.net
[3]http://poliqarp.sourceforge.net

```
'Bulgarian'  -b---z-aa--aanan-----------o-------c
'Belarusian' -z---o-----a-----zz----b-------b----
'Czech'      --------yyyyy--n------d-------z-----
'Croatian'   -b----ii-rrr---------z-----------o--
'Macedonian' -------------ynnz-------------------
'Polish'     y---nn-------nb----------yobb-------
'Russian'    -----o-bbaaa--------b---------------
'Slovak'     -o-b----y-aay----------------zz-a-y-
'Slovenian'  -----o-rrrrr--b-----------d---z-r-z-
'Serbian'    b-----i---rr-------z----z-------zd--
'Ukrainian'  --n--z--aaaa---n----z---z-zz---p----
```

Figure 2: Strings representing the word-aligned tokens as classified into types according to the user-supplied parameter file.

3.3 Aggregate Perspective

In a second perspective, the system outputs visualizations of the aggregate differences in distribution of the variables across different languages. As descibed above, tokens in the primary language and their equivalents are classified with respect to the user defined operationalization in terms of word, lemma, tag, and other possible levels, just as it is done for the random samples. The examples are thus converted to strings as shown in figure 2. If the strings are seen as a table, each column represents a set of word-aligned word forms, with each word form represented as a single letter reflecting the type it was classified as.

The functional similarity or dissimilarity between the distribution of the variables in multilingual versions of the same text is then computed by determining for each pair of texts the overlap in the occurrences of this variable in aligned tokens, i.e.:

$$dist(V_{lng1}, V_{lng2}) = 1 - \frac{V_{lng1} \cap V_{lng2}}{V_{lng1} \cup V_{lng2}}$$

In other words, the system computes strings of word-aligned corpus positions that are labeled according to the classification in the parameter file and computes the hamming distance between these strings. This computation is used to arrive at distance matrices describing the similarity or dissimilarity of the distribution of the variable between texts. These matrices are visualized in NeighborNets (Huson and Bryant, 2006), a clustering algorithm that was chosen since it preserves much of the ambiguity we find in this data.

Using different filters and definitions of the features that are being compared, the system can then be used to output different visualizations of the differences in distribution of the variables in question. This concerns (a) the distances of texts/languages to each other with respect to some variable (see fig. 3); (b) the distances of the variables to each other taken as cross-linguistic types; this is calculated for each pair of variables by determining the proportion of examples where both are used in equivalent word forms (see fig. 4); (c) the distance of the language specific instantiations of the variables to each other; this allows to see whether formants of different classes overlap in their domain (see fig. 5). In addition, it outputs documents with word lists of equivalent tokens in different languages and their classification (not shown here).

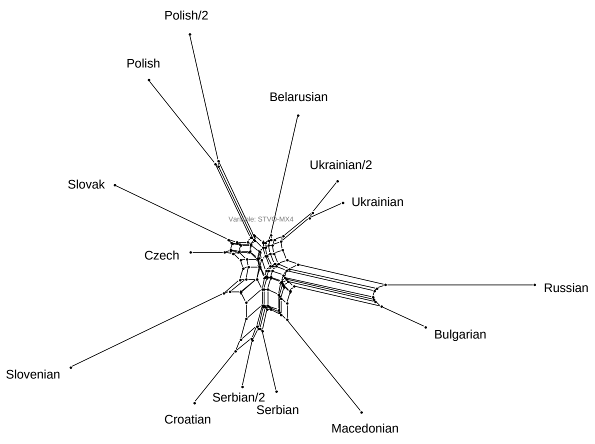


Figure 3: Similarity of use of nouns derived with the suffix class OST in multiple versions of the same text in different Slavic languages

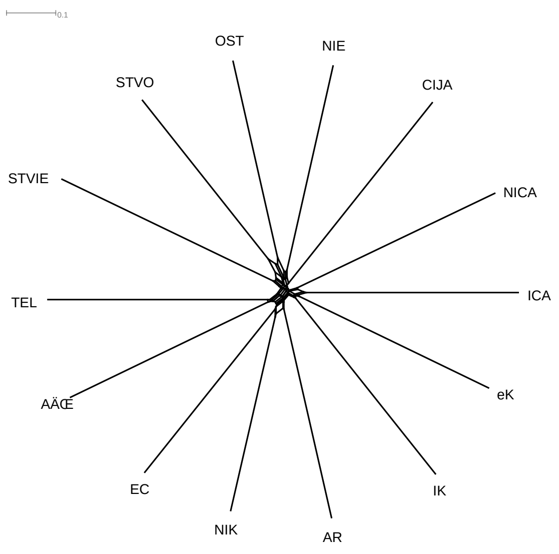


Figure 4: Suffix classes across Slavic: functional similarity.

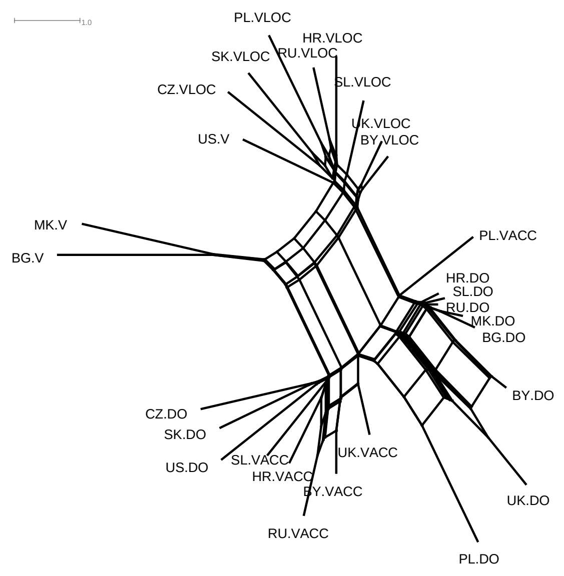


Figure 5: Functional similarity of Slavic prepositions (language specific representations)

4 Concluding Remarks

I have presented a component that supplements a query-based interface to a word-aligned multilingual parallel corpus with a new comparative corpus evaluation component which is available offline and is planned to be implemented as a web service. This corpus evaluation component will provide users with the possibility to upload their own parameter files which provide complex definitions of comparable items in different languages based on their formal characteristics. The corpus is then evaluated in respect to the functional similarity of the items in question. Crucially, the component aims to give both an aggregate and a detailed view of the data, so that the user keeps the possibility to interpret the aggregate picture, and refine the parametrization as necessary for his or her needs.

Acknowledgments

I gratefully acknowledge funding by the Swiss National Science Foundation, grant 151230 *Convergence and divergence of Slavic from a usage based, parallel corpus driven perspective*.

References

František Čermák and Alexandr Rosen. 2012. The case of InterCorp, a multilingual parallel corpus. *International Journal of Corpus Linguistics*, 13(3):411–427.

Michael Cysouw and Bernhard Wälchli, editors. 2007. *Parallel Texts: Using translational equivalents in linguistic typology. Special Issue of STUF 60/2.*

Östen Dahl. 2014. The perfect map: Investigating the cross-linguistic distribution of tame categories in a parallel corpus. In Benedikt Szmrecsanyi and Bernhard Wälchli, editors, *Aggregating Dialectology and Typology: Linguistic Variation in Text and Speech, within and across Languages*, pages 268–289. De Gruyter Mouton, Berlin, New York.

Stefan Evert and Andrew Hardie. 2011. Twenty-first century corpus workbench: Updating a query architecture for the new millennium. In *Proceedings of the Corpus Linguistics 2011 Conference, Birmingham, UK*. University of Birmingham.

Daniel H. Huson and David Bryant. 2006. Application of phylogenetic networks in evolutionary studies. *Mol. Biol. Evol.*, 23:254–267.

Roland Meyer, Ruprecht von Waldenfels, and Andreas Zeman. 2006-2014. Paravoz - a simple web interface for querying parallel corpora. https://bitbucket.org/rvwfels/paravoz.

Pavel Rychlý. 2007. Manatee/bonito - a modular corpus manager. In *1st Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 65–70, Brno. Masaryk University.

Jörg Tiedemann. 2003. *Recycling Translations – Extraction of Lexical Data from Parallel Corpora and their Application in Natural Language Processing*. Ph.D. thesis, Uppsala University, Uppsala, Sweden. Anna Sågvall Hein, Åke Viberg (eds): Studia Linguistica Upsaliensia.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Martin Volk, Johannes Graën, and Elena Callegaro. 2014. Innovations in parallel corpus search tools. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ruprecht von Waldenfels. 2014. Explorations into variation across Slavic: taking a bottom-up approach. In Benedikt Szmrecsanyi and Bernhard Wälchli, editors, *Aggregating Dialectology and Typology: Linguistic Variation in Text and Speech, within and across Languages*, pages 290–323. De Gruyter Mouton, Berlin, New York.