LREC 2018 Workshop

Challenges in the Management of Large Corpora (CMLC-6)

PROCEEDINGS

Edited by

Piotr Bański, Marc Kupietz, Adrien Barbaresi,
Hanno Biber, Evelyn Breiteneder, Simon Clematide, Andreas Witt

ISBN: 979-10-95546-14-6
EAN: 9791095546146

7 May 2018

Proceedings of the LREC 2018 Workshop
"Challenges in the Management of Large Corpora (CMLC-6)"

07 May 2018 – Miyazaki, Japan

Edited by Piotr Bański, Marc Kupietz, Adrien Barbaresi,
Hanno Biber, Evelyn Breiteneder, Simon Clematide, Andreas Witt

http://corpora.ids-mannheim.de/cmlc-2018.html

Organising Committee

- Piotr Bański (IDS Mannheim)
- Marc Kupietz (IDS Mannheim)
- Adrien Barbaresi (Academy Corpora, Austrian Academy of Sciences)
- Stefan Evert (Friedrich-Alexander-Universität Nürnberg/Erlangen)
- Hanno Biber (Academy Corpora, Austrian Academy of Sciences)
- Evelyn Breiteneder (Academy Corpora, Austrian Academy of Sciences)
- Simon Clematide (Institute of Computational Linguistics, UZH)
- Andreas Witt (University of Cologne, IDS Mannheim, University of Heidelberg)

Programme Committee

- Vladimír Benko (Slovak Academy of Sciences)
- Felix Bildhauer (IDS Mannheim)
- Hennie Brugman (Meertens Institute, Amsterdam)
- Steve Cassidy (Macquarie University)
- Dan Cristea ("Alexandru Ioan Cuza" University of Iasi)
- Damir Ćavar (Indiana University, Bloomington)
- Tomaž Erjavec (Jožef Stefan Institute)
- Stefan Evert (Friedrich-Alexander-Universität Nürnberg/Erlangen)
- Alexander Geyken (Berlin-Brandenburgische Akademie der Wissenschaften)
- Andrew Hardie (Lancaster University)
- Serge Heiden (ENS de Lyon)
- Nancy Ide (Vassar College)
- Miloš Jakubíček (Lexical Computing Ltd.)
- Dawn Knight (Cardiff University, UK)
- Michal Křen (Charles University, Prague)
- Sandra Kübler (Indiana University, Bloomington)
- Krister Lindén (University of Helsinki)
- Anke Lüdeling (HU Berlin)
- Uwe Quasthoff (Leipzig University)
- Paul Rayson (Lancaster University)
- Martin Reynaert (Tilburg University)
- Laurent Romary (INRIA)
- Roland Schäfer (FU Berlin)
- Roman Schneider (IDS Mannheim)
- Serge Sharoff (University of Leeds)
- Ludovic Tanguy (University of Toulouse)
- Dan Tufiş (Romanian Academy, Bucharest)
- Tamás Váradi (Research Institute for Linguistics, Hungarian Academy of Sciences)
- Pavel Vondřička (Charles University, Prague)
- Amir Zeldes (Georgetown University)

Introduction

Large corpora require careful design, licensing, collecting, cleaning, encoding, annotation, management, storage, retrieval, analysis, and curation to unfold their potential for a wide range of research questions and users, across a number of disciplines. Apart from the usual CMLC topics that fall into these areas, the 6th edition of the CMLC workshop features a special focus on corpus query and analysis systems and specifically on goals concerning their interoperability.

In the past 5 years, a whole new generation of corpus query engines that overcome limitations on the number of tokens and annotation layers has started to emerge at several research centers. While there seems to be a consensus that there can be no single corpus tool that fulfills the need of all communities and that a degree of heterogeneity is required, the time seems ripe to discuss whether (further, unrestricted) divergence should be avoided in order to allow for some interoperability and reusability – and how this can be achieved. The two most prominent areas where interoperability seems highly desirable are query languages and software components for corpus analysis. The former issue is already partially addressed by the proposed ISO standard Corpus Query Lingua Franca (CQLF). Components for corpus analysis and further processing of results (e.g. for visualization), on the other hand, should in an ideal world be exchangeable and reusable across different platforms, not only to avoid redundancies, but also to foster replicability and a canonization of methodology in NLP and corpus linguistics.

The 6th edition of the workshop is meant to address these issues, notably by including an expert panel discussion with representatives of tool development teams and power users.

P. Bański, M. Kupietz, A. Barbaresi, H. Biber, E. Breiteneder, S. Clematide, A. Witt May 2018

Programme

Session 1: Management and Search

09.00 – 09.10 Piotr Bański, Adrien Barbaresi, Marc Kupietz and Andreas Witt
Opening

09.10 – 09.30 Christoph Kuras, Thomas Eckart, Uwe Quasthoff and Dirk Goldhahn
Automation, Management and Improvement of Text Corpus Production

09.30 – 09.50 Thomas Krause, Ulf Leser, Anke Lüdeling and Stephan Druskat
Designing a Re-Usable and Embeddable Corpus Search Library

09.50 – 10.10 Radoslav Rábara, Pavel Rychlý and Ondřej Herman
Distributed Corpus Search

10.10 – 10.30 Adrien Barbaresi and Antonio Ruiz Tinoco
Using Elasticsearch for Linguistic Analysis of Tweets in Time and Space

10.30 – 11.00 **Coffee Break**

Session 2: Query and Interoperability

11.00 – 11.20 Marc Kupietz, Nils Diewald and Peter Fankhauser
How to Get the Computation Near the Data: Improving Data Accessibility to, and Reusability of Analysis Functions in Corpus Query Platforms

11.20 – 11.40 Roman Schneider
Example-Based Querying for Specialist Corpora

11.40 – 12.00 Paul Rayson
Increasing Interoperability for Embedding Corpus Annotation Pipelines in Wmatrix and Other Corpus Retrieval Tools

12.00 – 13.00 **Panel Discussion:**
Interoperability and Extensibility of Analysis Components in Corpus Query Tools

Table of Contents

Automation, Management and Improvement of Text Corpus Production

Christoph Kuras, Thomas Eckart, Uwe Quasthoff, Dirk Goldhahn
University of Leipzig
Augustusplatz 10, 04109 Leipzig
{ckuras,teckart,quasthoff,dgoldhahn}@informatik.uni-leipzig.de

Abstract

The process of creating large text corpora for different languages, genres, and purposes from data available on the Web involves many different tools, configurations, and – sometimes – complex distributed hardware setups. This results in increasingly complex processes with a variety of potential configurations and error sources for each involved tool. In the field of commercial management, Business Process Management (BPM) is used successfully to cope with similar complex workflows in a multi-actor environment. Like enterprises, research environments are facing a gap between the IT and other departments that needs to be bridged and also have to adapt to new research questions quickly. In this paper we demonstrate the usefulness of applying these approved strategies and tools to the field of linguistic resource creation and management. For this purpose an established workflow for the creation of Web corpora was adapted and integrated into a popular BPM tool and the immediate benefits for fault detection, quality management and support of distinct roles in the generation process are explained.

Keywords: corpus creation, process management, scientific workflows, BPM

1. Challenges of large-scale Text Corpus Production

Creating large text corpora for many different languages involves executing an extensive set of applications in a – more or less – defined order. This includes applications for preprocessing and annotation starting from sentence segmentation through to various annotation tools like part-of-speech taggers or parsers. For different kinds of text material and different languages there are typically varying configurations for each of these applications. Furthermore, the selection of applied tools might differ depending on the input material's language or language family as it is the case for special forms of word tokenization approaches. All in all this results in complex chains of tools with a variety of possible configurations.

The resulting solution has to be seen in the context of conflicting requirements: a systematic corpus production process has to be streamlined and automated in order to keep up with ongoing data acquisition, which may - in extreme cases - comprise minute-wise updates for news material or content obtained from social networking services. On the other hand, in collaboration with other researchers, new research questions arise continuously, making it crucial to be as flexible as possible when it comes to adaptions in the workflow. Another relevant aspect is the ability to trace errors in the running workflow. When executing a complex chain of applications, identifying errors in any of the processing steps can be time consuming especially if (partial) results are examined manually. As a consequence a systematic approach is needed to document configurations for every single execution of each application. Combined with an automatic data sampling many kinds of problems might be recognized, so processes can be interrupted already in an early stage to take any actions necessary. A thorough documentation of applied criteria also ensures the reproducibility of results; additionally the data can be used in terms of fault tracing.

Even more problem areas evolve when multiple persons or organizational units are involved in the creation process. This is for example the case when computing power is outsourced to commercial companies or when an external group of experts is in charge of reviewing or annotating data resulting from one of the processing steps. These external dependencies result in an even more complex process. This may lead – if not controlled and monitored properly – to inefficiencies due to disruptions in the process flow and becomes even more important when resources are processed in parallel. As a result, there is a demand for a system controlling the overall process execution and monitoring specific metrics which can be used to forecast execution time, allow statements about error rates at different steps, and similar issues.

All these aspects are motivations for modeling and executing scientific workflows in Natural Language Processing (NLP). The existence of a variety of approaches, reaching from NLP-related tools like GATE through data analysis software like RapidMiner to supporting tools for managing scientific workflows like Apache Taverna or Kepler[1] illustrates the pressing demand in this area. In fact, a process-oriented view supported by powerful applications is already present in the field of economics for a long time. Business Process Management (BPM) (Aalst et al., 2000; vom Brocke et al., 2010; Aalst et al., 2016; Hirzel et al., 2013) has become extremely popular in commercial contexts and many of its features make it also useful for NLP-related tasks in a complex NLP environment.

2. Managing Corpus Creation with a Workflow-Management System

There are many tools available that can be used to model processes and that even allow to execute them. However, some of them solely model a data-centered view[2], describing how the data should be transformed, often including technical aspects of the respective implementation. These

[1] Which rely on a very generic definition for the term "scientific workflow": "an executable representation of the steps required to generate results.".

[2] Like RapidMiner or the Konstanz Information Miner.

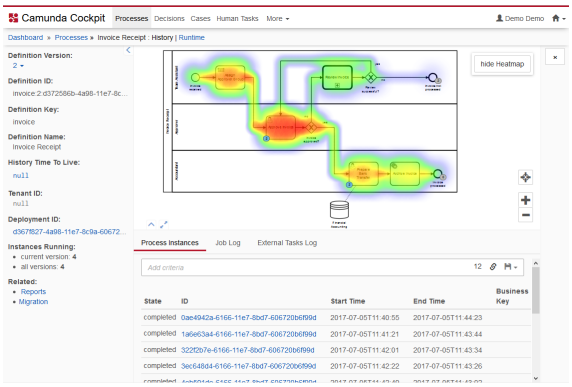


Figure 1: Process heatmap in *Camunda Cockpit*

tools mostly define a model that includes operators shipping with that specific software. Other approaches, like Apache Taverna, are able to orchestrate different types of scripts and web services that can be hosted at remote locations. However, these applications are not capable of modeling the aspect of collaboration between organizations arising from the integration of hardware at different organizational units or even human beings. Moreover, many parallels can be drawn between the support of IT regarding research questions and the support of IT in business scenarios (cf. Kuras and Eckart (2017)). For that reason it seems natural to apply some of the strategies originating in that field. One of these strategies is Business Process Management (BPM). This is a management approach which is mainly targeted at streamlining processes within an organization and making the business more flexible, making it able to adapt to changes in the market quickly. One of the standard ways to model processes in this field is the Business Process Model and Notation (BPMN) which is a standard of the Object Management Group[3], currently in version 2 released in 2011 (OMG (2011)). Modeling in BPMN has the advantage that these models can be enhanced by technical specifications making it possible to execute them directly within a workflow management system (cf. Gadatsch (2012)). Popular solutions include jBPM, Activiti and Camunda.

For managing the considered corpus building process, the Camunda workflow management system is used[4]. The reasons for this decision lie especially in its open availability[5], its utilization in a variety of – often commercial – scenarios, and the availability of different helpful extensions and user-friendly interfaces. However, the use of BPMN as primary means of describing and executing workflows is not bound to a specific workflow management system; other software solutions could have been used instead.
Figure 1 shows a screenshot of the web interface to the Camunda process execution engine[6]. The system monitors the runtimes of each process instance and each task within this execution. This enables to generate a heatmap overlay revealing possible bottlenecks in the process by marking the tasks in which the process spends most of the execution time in average (red). This is a functionality that can be used only by monitoring the runtimes which is done by default. When modeling a process in Natural Language Processing, many different measures, which are called Key Performance Indicators (KPI) in the field of BPM, can be imagined (cf. Gadatsch (2012)). These measures can not only be used to monitor, streamline and improve the process itself but to ensure the quality of data being generated during the runtime of the process (see Section 4.).

3. Distributed Corpus Creation at the LCC

The Leipzig Corpora Collection (LCC) (Goldhahn et al., 2012) continuously generates corpora for a large number of languages. As more and more text material becomes available through the Web, massively-parallel crawling can result in amounts of raw data in the range of hundreds of gigabytes[7] that have the potential to pile up to almost unprocessable "data heaps".
To handle these amounts of data in an acceptable period of time the already established processing workflow was extended by integrating an external computing center. This computing center provides a high-performance computing (HPC) cluster via a RESTful API using the UNICORE interface (Uniform Interface to Computing Resources[8]). The overhead of delegating working steps to an external computing facility consists here mostly of data transfer times and is in the current configuration – at least compared to the actual data processing – rather slim.
Figure 2 depicts the coarse model of the process in BPMN notation. In the first step, available raw data is selected, which then gets preprocessed using the resources of the external partner. After that, the data is enriched locally by the calculation of cooccurrences and finally imported into a relational database.
An important aspect of BPMN is the possibility to model subprocesses hiding complexity. On one hand this enables personnel not familiar with the process to quickly get an overview. On the other hand, expected faults of the process execution can be modeled directly. This enables the process engine to decide which actions have to be taken in case of an error, making the execution even more efficient by saving the costs of additional human interactions. Figure 3 shows a more detailed variant of the preprocessing subprocess. It basically consists of these steps:

- *sentence segmentation*: segment raw text data into sentences
- *sentence cleaning*: remove sentences that are, based on patterns, undesirable

[3] https://www.omg.org
[4] https://camunda.org
[5] The Camunda platform is licensed under the Apache License 2.0.
[6] https://docs.camunda.org/manual/7.7/webapps/cockpit/bpmn/process-history-views/
[7] 4 Terabytes of incoming raw material per day are typical values for LCC crawling processes.
[8] https://www.unicore.eu/

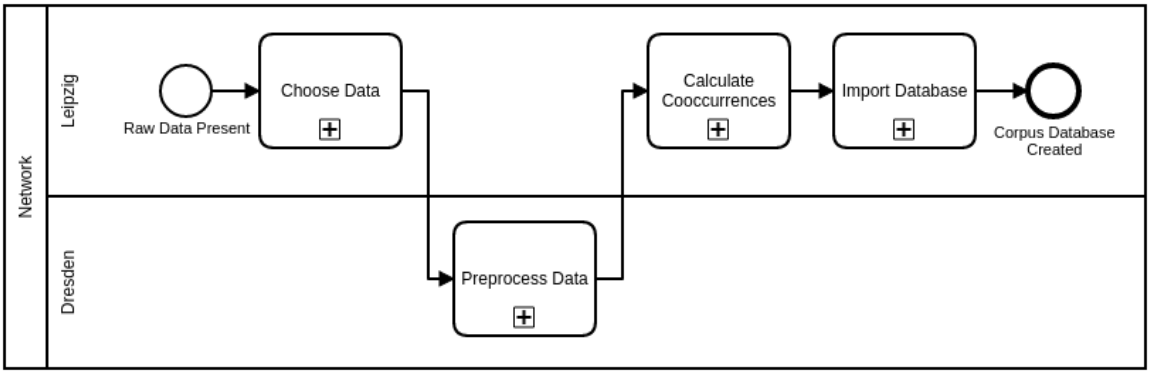


Figure 2: Coarse model of the corpus production steps

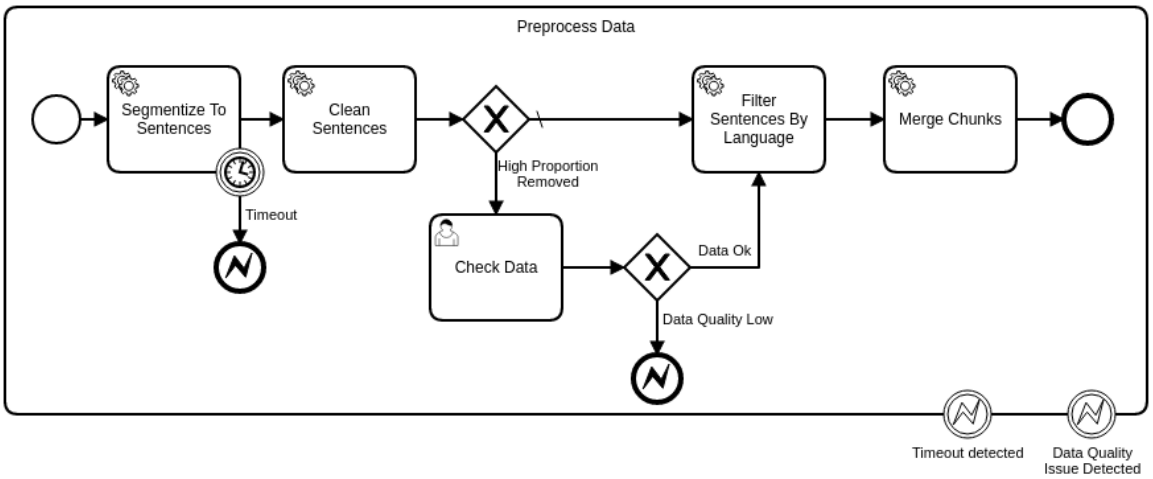


Figure 3: More elaborate model of the preprocessing subprocess

- *language filtering*: filter out sentences not belonging to the target language
- *merging*: merge all chunks of sentences (due to parallel processing)

To illustrate the possibilities of BPMN modeling the subprocess is enhanced by different mechanisms to detect fault during the process execution. An *intermediate boundary timer event* is added to the *Segmentize To Sentences* task. If the task exceeds a defined time limit, an *error end event* will be thrown. On the subprocess border, a *boundary error catching event* for the error kind "Timeout" is installed. This will ensure that all thrown "Timeout" events from within the subprocess will be caught and can then be handled outside the subprocess. After cleaning the sentences by removing lines matching predefined patterns (e.g. sentences containing too many digits or that are excessively long), a decision with respect to the sequence flow is modeled. Based on the available process data, the execution engine automatically selects the path to execute by evaluating a predefined condition[9]. At this point, the process engine will choose the path depending on the proportion of removed sentences. The decision is based on the measuring data available during execution; the exact calculation has to be specified in the model. In case the proportion is above a predefined threshold, a *User Task* will be assigned to a human actor who has access to the Camunda web interface. The user is automatically informed about the assignment of the new task; it is also possible to assign a task to a group of authorized users from which one person can then claim the task. A small set of randomly sampled sentences can be presented to the user who will then decide whether there is a general issue with the quality of the underlying data. According to the user decision, the process will throw a data quality related error or continue the process execution normally. The actual error handling may include a message to personnel responsible for the input data.

[9]The decision is modeled using a XOR-Gateway.

4. Measurement and Improvement of Corpus Quality

One of the main purposes of BPM is the continuous improvement and quality assurance of the managed processes (cf. Reichert and Lohrmann (2010)). This is also an important aspect concerning the production of high quality lin-

guistic resources. The area of fault detection and quality assurance can be seen and handled with a focus on a variety of criteria. This includes setting the evaluation focus on the process itself or on the actual results of every intermediate step and the final results. Both require a systematic monitoring and appropriate procedures if deficits were identified. Typical criteria for the first evaluation – focused on rather "technical" indicators – include, for example, basic information if processes or subprocesses terminated unexpectedly, the extent of technical resources that were consumed, or technical constraints that were or were not met (including weak or strong temporal constraints).

A key aim of such a thoroughly monitored environment is the extraction and identification of process patterns. Over time, expectancy values for all metrics evolve so that a problematic process instance can be revealed at the earliest possible moment[10]. Based on a set of predefined rules this allows the workflow management system to decide automatically whether to cancel the process, saving processing time and resources. Furthermore, these metrics can be used to make statements about the quality of the overall outcome right after the processing has finished, for example how "noisy" a corpus is or how it compares with similar resources. This does not only apply to the process as a whole but to each single task involved in the process, making it easier and faster to spot problems during execution, which is especially important when executions run for long periods of time. Manually performed checks, though feasible with smaller data and less frequent executions, would be error-prone and inefficient with respect to an "industrial-scale" corpus production that is done within the context of the LCC. Supporting and controlling processes with a workflow management system ensures the systematic application and evaluation of all relevant criteria.

These criteria are a typical starting point for the identification of general problems or performance issues like the identification of bottlenecks, implementation inefficiencies or alike. As the processing of big data material often has to deal with performance issues and the efficient usage of available hardware, optimizing the structure of those processes is of special interest. This also requires a detailed recording of the processes' tasks runtimes and latency times for a larger – representative – number of executions. Parallels to other disciplines are therefore hardly surprising: many standard metrics in the field of logistics apply for the "logistics" of NLP pipelines as well and can be used as inspiration, e.g. analysing historical data being able to forecast trends concerning future resource needs like storage and processing power (cf. Robinson (1989)). Data storage can be seen as a stock, especially when multiple storages are needed due to the integration of remote computing centers and the data needs to be transferred from one stock to another[11]. Avoiding supply shortages as well as excess stocks, e.g. as a result of an imbalance between data collection and processing time, is crucial for an efficient use of resources during the processing of large corpora. The creation of text corpora is an end-to-end process that reaches from the collection of raw data through to the delivery to an end user, thus can also be considered as a supply chain that involves many suppliers of data, providers of services and end users demanding the actual outcome of the process.

A less technical viewpoint focuses on the actual outcomes of a process, which is data as the result of a sequence of subprocesses. Quality of data is typically related to its usability for a specific purpose and hence, often hard to measure automatically. However, even simple and easily determinable criteria may function as useful indicators. In the case of NLP tools this may be the comparison of input material to output material sizes (like the amount of raw text with the resulting number of sentences, types, or tokens), checking the completeness of different annotation layers, or identifying untypical distributions of annotations for the language family, language, source or genre in question. Additionally, any linguistic invariant may function as a "deep" indicator for the well-formedness of language material, especially in cases where input data is of uncertain quality. As this often requires specific annotations or even human intuition, simple principles based on language statistics may sometimes suffice as a substitute (Eckart et al., 2012). In any case, BPM allows their integration and more checks as an integral component of the execution and evaluation of every process instance. Furthermore, it provides build-in capabilities to support both automatic and manual checks. These data not only can increase the quality of the outcome of a single process instance based on automatic sequence-flow decisions. It can also build the foundation for the analysis of historical process data allowing assertions about the performance of subprocesses and the consumption of resources. For instance, recorded process data in a test case revealed the proportions of average subprocess time consumptions to be 34% for preprocessing, 64% for the calculation of cooccurrences and 1% for the database creation. As more data are collected, more sophisticated statements about the impact of text type and data size on these measures are possible. Historical data can also be used to spot configuration problems concerning specific languages. Regarding the loss of size after filtering the sentences by the target language, a size reduction ranging from 3% to 18% was observed in a test case, depending on language and origin of the data. However, such deviations can also indicate configuration problems. Another measurable aspect is the throughput of data of the involved services which may also reveal patterns pointing towards configuration problems or quality issues with resources used by these services. Furthermore, this allows forecasting of runtimes, again with respect to language and origin of the data, making the prediction more reliable. As some checks require profound knowledge in specific fields, e.g. linguistics, manual checks might still be necessary. With an appropriate process model and a workflow management system, even such manual checks, often completely isolated from fully-automated tasks, can be integrated into the process execution. Inspections by domain or language experts may function as prerequisite for the publication of a resource. Comparable to other workflows in a highly specialized working environment this requires a consistent rights

[10] Those values are of course specific for different characteristics of the input material like its language, source format or origin.

[11] In such computing centers, high performance data storage can be strictly limited.

management and the assignment of accurate process roles.

5. Summary

In contrast to many "proprietary" solutions that are often used in practice, BPMN is a well known and documented standard that supports a common understanding of processes, interfaces and interrelations for all involved participants. It can be used as a standardized description of a workflow and is at the same time executable in different workflow management solutions. Being an established standard, it has the benefit of support by different software tools and simplifies reuse in other contexts.

Both the techniques and the tools used in Business Process Management prove to be useful in the process of the repeated production of large corpora. BPM allows the definition of complex processes using a heterogeneous infrastructure for different corpus processing tools and intermediate human interaction. This allows an embedded quality control for the data which are processed. In the case of adaptations of the corpus creation process (for instance, with a special word tokenization tool for a new language), this can be modeled transparently. Furthermore, the approach is highly flexible as it allows reusing tasks in new process models or extending fully automated processes by human interaction without having to modify task implementations. In addition to that, replacing underlying implementations of tasks is possible without changing the process itself. The model is non-technical and understandable for non-programmers; parameters like runtime distribution can be visualized user-friendly for process monitoring.

6. Bibliographical References

Aalst, W. v. d., Desel, J., and Oberweis, A. (2000). *Business Process Management Models, Techniques, and Empirical Studies*. Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg.

Aalst, W. v. d., La Rosa, M., and Santoro, F. M. (2016). Business process management. *Business & Information Systems Engineering*, 58(1):1–6, Feb.

Eckart, T., Quasthoff, U., and Goldhahn, D. (2012). Language Statistics-Based Quality Assurance for Large Corpora. In *Proceedings of Asia Pacific Corpus Linguistics Conference 2012, Auckland, New Zealand*.

Gadatsch, A. (2012). *Grundkurs Geschäftsprozess-Management - Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker*. Springer Vieweg, 7. edition.

Goldhahn, D., Eckart, T., and Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*, pages 759–765.

Hirzel, M., Geiser, U., and Gaida, I. (2013). *Prozessmanagement in der Praxis - Wertschöpfungsketten planen, optimieren und erfolgreich steuern*. Springer Gabler, 3. edition.

Kuras, C. and Eckart, T. (2017). Prozessmodellierung mittels BPMN in Forschungsinfrastrukturen der Digital Humanities. In *INFORMATIK 2017, Lecture Notes in Informatics (LNI)*.

OMG. (2011). Business process model and notation BPMN - Version 2.0. `http://www.omg.org/spec/BPMN/2.0/PDF`, retrieved 2017-09-28.

Reichert, M. and Lohrmann, M. J. (2010). *Basic considerations on business process quality*. Ulmer Informatik-Berichte. Universität Ulm. Fakultät für Ingenieurwissenschaften und Informatik, Ulm.

Robinson, D. (1989). IT in Logistics. *Logistics Information Management*, 2(4):181–183.

vom Brocke, J., Rosemann, M., and Bernus, P. (2010). *Handbook on Business Process Management*. International Handbooks on Information Systems. Springer, Heidelberg u.a.

Designing a Re-Usable and Embeddable Corpus Search Library

Thomas Krause[1], Ulf Leser[2], Anke Lüdeling[1], Stephan Druskat[1]
[1]Humboldt-Universität zu Berlin, Dept. of German Studies and Linguistics, Unter den Linden 6, D-10099 Berlin
[2]Humboldt-Universität zu Berlin, Dept. of Computer Science, Unter den Linden 6, D-10099 Berlin
krauseto@hu-berlin.de, leser@informatik.hu-berlin.de, {anke.luedeling, stephan.druskat}@hu-berlin.de

Abstract

This paper describes a fundamental re-design and extension of the existing general multi-layer corpus search tool ANNIS, which simplifies its re-use in other tools. This embeddable corpus search library is called graphANNIS and uses annotation graphs as its internal data model. It has a modular design, where each graph component can be implemented by a so-called graph storage and allows efficient reachability queries on each graph component. We show that using different implementations for different types of graphs is much more efficient than relying on a single strategy. Our approach unites the interoperable data model of a directed graph with adaptable and efficient implementations. We argue that graphANNIS can be a valuable building block for applications that need to embed some kind of search functionality on linguistically annotated corpora. Examples are annotation editors that need a search component to support agile corpus creation. The adaptability of graphANNIS and its ability to support new kinds of annotation structures efficiently could make such a re-use easier to achieve.

Keywords: corpus tools, multi-layer corpora, interoperability, graph database

1. Introduction

The creation of a search tool for linguistic corpora can be a large development effort. Nevertheless, new tools are still being developed, due to their necessity for studies in corpus linguistics as well as the increasing diversity of corpus types and phenomena available for study.[1] Exemplary reasons for such a re-development are:

(1) new kind of annotation structures not supported by any other tool,

(2) non-ergonomic integration of the existing tool,

(3) problematic software licenses, or

(4) performance problems with the existing tool.

Problem (1) holds true especially true for multi-layer corpora (Dipper, 2005), where different kinds of annotations are combined into the same corpus and where it can be expected that the corpus is extended with new types of annotation over time. Multi-layer corpus search tools like ANNIS (Krause and Zeldes, 2016) are often designed to support many kinds of annotation structures generically in the same software and query language. This generality in ANNIS is accomplished by using annotation graphs (Bird and Liberman, 2001) as the underlying data model. Despite the general data model, there are corpora which are difficult to represent in ANNIS, e.g., corpora for sign language, which need support for more complex concepts of tokens, temporal and spatial annotations (Hanke and Storz, 2008). Thus, even a generic multi-layer corpus search tool like ANNIS needs to be extended regularly to support more types of data. The problems (2) and (3) can occur if a non-search-centric tool, such as an annotation editor, needs to be extended with search functionality, e.g., because it is supposed to support an agile corpus creation workflow (Voormann and Gut, 2008; Druskat et al., 2017). Also, performance issues (4) often prove to be a permanent problem. While computer hardware evolves, the need for larger corpora with more tokens, more annotation layers and more relationships (even between documents if text reuse is studied (Berti et al., 2014)) increases as well, and keeping up with the amount of data poses a constant challenge.

When the need for a new corpus tool or integration of a query system in an existing software arises, it can be more sustainable to at least partially rely on an existing solution. Consider the following example: An annotation tool needs a custom user interface for presenting search results with tight integration to an existing editor. It would not necessarily need to implement a new query language or query engine, or a whole query system with all necessary components. Instead, it could simply re-use parts of existing domain-specific query systems. This paper describes a fundamental re-design and extension of the existing general multi-layer corpus search tool ANNIS, which simplifies its re-use in other tools.

2. graphANNIS

We want to present an approach to design an embeddable corpus search library that addresses the aforementioned problems, and discuss the benefits and downsides of this design. The library that will be used as a case study is the graphANNIS query engine, which is described in more detail in Krause et al. (2016). graphANNIS was used to re-implement the ANNIS Query Language (AQL) (Rosenfeld, 2010; Krause and Zeldes, 2016) as a main memory query engine in the C++ programming language. It can represent the same range of annotation types as the original ANNIS implementation, which includes

- token annotations and multiple tokenization,
- span annotations,

[1]Many dedicated and general search tools exists. Examples for generic search tools are CWB (Evert and Hardie, 2011), KorAP (Diewald and Margaretha, 2016), TIGERSearch (Lezius, 2002) or EXMARaLDA EXAKT (Schmidt and Wörner, 2014).

- dominance relations (e.g. for syntax trees), and
- pointing relations for generic edges between any annotation node.

While it supports AQL, the implementation is much more flexible compared to the original mapping of AQL to SQL. It uses directed graphs that are partitioned into acyclic components as its basic data model.
GraphAnnis holds all data in memory, which poses limits on the sizes of corpora that can be queried. In Krause et al. (2016), the used memory for different kind of corpora is reported. The "Parlamentsreden_Deutscher_Bundestag" corpus (Odebrecht, 2012) contains around 3.1 million tokens and each token has part-of-speech and lemma annotations. It uses less than 600 MB of main memory. Servers with 512 GB of main memory are available, and such a server could host corpora $\sim$ 850 times larger than the "Parlamentsreden_Deutscher_Bundestag" corpus. For such simple token-only corpora, the memory consumption of graphANNIS raises linearly with the number of tokens. Thus, even corpora with approximately 2.6 billion tokens, including part-of-speech and lemma annotation, should fit into the main memory of such a server. The size of a corpus cannot ultimately be defined by numbers of tokens alone. Instead, a depth factor must be taken into account, referring to the complexity of a corpus, that is, the numbers of annotation nodes and edges on top of tokens, and across different annotation layers. Deeply annotated corpora with a large number of layers and consequently a large number of nodes and/or relations can arguably be defined as "large".

3. Modular implementation

Each component of the annotation graph can be implemented in graphANNIS in a specialized module (see Figure 1 for an overview). Such a graph storage is optimized to find reachable nodes and distances between nodes inside a component. In contrast to the similar Ziggurat design (Evert and Hardie, 2015), the implementation optimization is agnostic to the type of annotation that is encoded in this graph and only depends on the graph structure itself. Query operators can be implemented by using the reachability and distance functions of these components to efficiently implement queries for linguistic annotation concepts like precedence or dominance. New operators can be added, which allows to add support for more query languages, or address currently missing features of AQL like the ones described in Frick et al. (2012).
GraphANNIS currently implements three different types of graph storages:

- one that stores the graph in an adjacency list and uses graph traversal for finding reachable nodes,
- a pre-/post-order based implementation based on the ideas of Grust et al. (2004), and
- a graph storage that stores linear graphs[2] by using a single order value per node.

[2]A Linear graph (or "path graph") is a tree where the maximum number of outgoing edges for a node is 1.

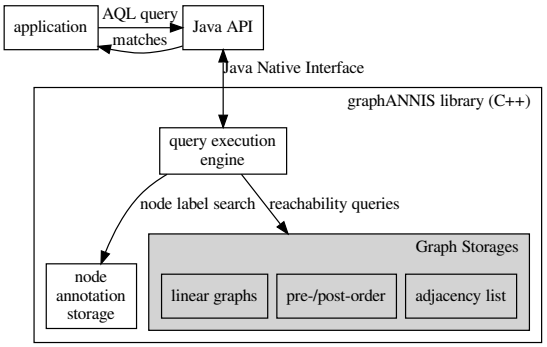


Figure 1: Overview of the graphANNIS library with its different graph storage modules.

The adjacency list is able to store all types of graph components, but the pre-/post-order based implementation is restricted to directed acyclic graphs and needs duplicate entries when the graph component is not a tree. When a corpus is imported, a heuristic is used to choose which graph storage best suits the given graph structure of each component. This modular design allows adding new types of graph storages when new annotations shall be supported by the query engine. These new graph storages can exploit the new types of graph structures and provide better performance than the existing implementations. For example, token precedence is modeled as explicit edges between tokens in graphANNIS. The length of the path between the first and the last token in the document is the number of tokens of the document. For queries that search for tokens that precede each other with indefinite length, a traversal on an adjacency list would be inefficient compared to direct lookup of an order value in a pre-/post-order encoding or the single order value of a linear graph. On the other hand, in cases where pre-/post-order encoding would result in duplicated entries because the annotation graph is not a tree, graph traversal can be more efficient instead.

4. Evaluation

In Krause et al. (2016), benchmarks have been executed to compare graphANNIS with the original relational database implementation of ANNIS. These benchmarks show that graphANNIS is around 40 times faster to execute a workload of more than 3,000 queries (collected from actual user interactions with the existing ANNIS system) from 17 corpora, than the relational database implementation. It could be argued, that a more monolithic graph-based implementation could handle the workload equally well. In order to test if our modular design and the specialized graph storages actually have a positive impact, additional benchmarks with a similar setup as in Krause et al. (2016), but on an updated set of queries, a have been performed.[3]

[3]We did not compare the performance with the relational database implementation in this paper because the focus is on the modularization. The data set including the queries will be released as part of a later publication, which will allow performing such a comparative benchmark.

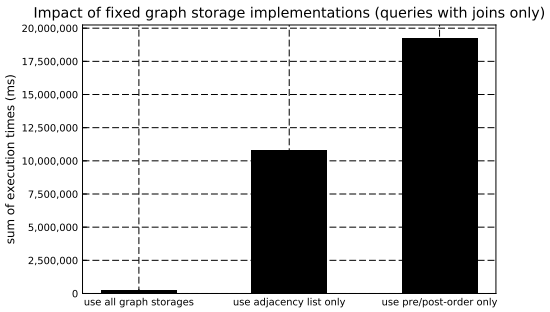


Figure 2: Impact of choosing different graph storage implementations per component on the execution time for the workload of 2,713 queries. Only queries that combine at least two annotation nodes haven been included in this workload. Overall performance is measured as the sum of the execution times for all queries in milliseconds. The baseline configuration (using all graph storages) executes the workload in $\approx 230,000$ ms.

The workload has been executed with the same version of graphANNIS on the same system, but with different optimization configurations. The configuration where all three graph storage implementations were enabled performed best (Figure 2). Another configuration, where the adjacency list graph storage has been used exclusively, was $>$40 times slower than the first one. Using graph indexes like the pre-/post-order alone does not help either, as this configuration is $>$80 times slower than the one with all graph storages enabled. There is no configuration where the graph storage for linear graphs has been used exclusively, because it is too specialized to represent all required graph types.
This experiment shows that the modularization actually has a positive effect. Our approach unites the interoperable data model of a directed graph with adaptable and efficient implementations. More detailed experiments to further investigate the strengths and potential problems of this approach are currently in progress.

5. Discussion and future work

While using a general data model is critical in supporting a wide range of possible annotation types, there are more practical issues that graphANNIS tries to solve as well. Reusing an existing query engine as software component can be challenging due to technical issues: If the system is only available as a web service for example, the developed software depends on the availability of this service, the network connection, and the sustainability of the infrastructure providing both. Even if the web service is open source, installing and managing it and all necessary dependencies (like a database management system) on a separate infrastructure can be overly complicated. graphANNIS is a software library that does not have dependencies that need to be installed separately. It is currently provided as both a C++ and a Java API. The library is also usable as OSGi[4] bundle, which made it possible to integrate it into the Atomic annotation tool (Druskat et al., 2014; Druskat et al., 2017).

[4] https://www.osgi.org/

Given the diverse landscape of corpus tools, providing only a C++ and a Java API does not seem sufficient. While a web service has the advantage of being indifferent to the programming language and operating system it is used by, an embedded software component can be more restricted in its ability to be integrated into these different types of systems. We therefore propose to extend graphANNIS with an API of the C programming language, which is supported by all major operating systems and programming languages. Such a C API would be a simplified interface to the functionality of graphANNIS and provide functions to execute queries (counting and finding instances), retrieving subgraphs, and administrative tasks like importing corpora. It would not expose the internal data structures like the actual graph storages.
As graphANNIS is available as an open source project licensed under the liberal Apache License, Version 2.0 on a public code hosting platform[5], the barriers for external contributors are already quite low. This measure also implies to foster sustainability and re-usability: when the integration into other programming languages or the feature set of graphANNIS is "almost sufficient" for an external project, it should be easier to extend graphANNIS than to develop a completely new system. GraphANNIS currently does not have some kind of dynamic plug-in system for extensions like adding new operators or graph storages, but changes by the community can be merged into the main code base. A possible problem for such a community-driven project is the usage of the C++ programming language, which is not widely used in other corpus linguistic projects. Extensive documentation, static code analysis, testing and continuous integration can help to make access easier and safer for new contributors. Alternatively, programming languages like Rust[6] provide more compile time guarantees for memory safety and absence of data races, with an execution efficiency similar to C++ and an easy way to provide an external C API. It should be evaluated if a port of graphANNIS to such a "safe" system programming language would be feasible and provide the same performance characteristics as the current C++ implementation. With the wide applicability of annotation graph data models for multi-layer corpora, efficient designs for graph search implementations on these models, techniques for easy integration into other tools and a community-driven development approach, the building blocks of a future interoperable linguistic query search library seem to be in sight and the development of such a system seems feasible.

6. Bibliographical References

Berti, M., Almas, B., Dubin, D., Franzini, G., Stoyanova, S., and Crane, G. R. (2014). The linked fragment: TEI and the encoding of text reuses of lost authors. *Journal of the Text Encoding Initiative*, (8).

Bird, S. and Liberman, M. (2001). A formal framework for linguistic annotation. *Speech Communication*, 33(1-2):23–60. Speech Annotation and Corpus Tools.

[5] https://github.com/thomaskrause/graphANNIS

[6] https://www.rust-lang.org

Diewald, N. and Margaretha, E. (2016). Krill: KorAP search and analysis engine. *JLCL*, 31(1):73–90.

Dipper, S. (2005). Xml-based stand-off representation and exploitation of multi-level linguistic annotation. In *Berliner XML Tage*, pages 39–50.

Druskat, S., Bierkandt, L., Gast, V., Rzymski, C., and Zipser, F. (2014). Atomic: An open-source software platform for multi-level corpus annotation. In *Proceedings of the 12th Konferenz zur Verarbeitung natürlicher Sprache (KONVENS 2014)*, volume 1, pages 228–234, Hildesheim.

Druskat, S., Krause, T., Odebrecht, C., and Zipser, F. (2017). Agile creation of multi-layer corpora with corpus-tools.org. In *DGfS-CL Poster Session. 39. Jahrestagung der Deutschen Gesellschaft für Sprachwissenschaft (DGfS)*, Saarbrücken, March.

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

Evert, S. and Hardie, A. (2015). Ziggurat: A new data model and indexing format for large annotated text corpora. *Challenges in the Management of Large Corpora (CMLC-3)*, page 21.

Frick, E., Schnober, C., and Banski, P. (2012). Evaluating query languages for a corpus processing system. In *LREC*, pages 2286–2294.

Grust, T., Keulen, M. V., and Teubner, J. (2004). Accelerating XPath evaluation in any RDBMS. *ACM Transactions on Database Systems (TODS)*, 29(1):91–131.

Hanke, T. and Storz, J. (2008). ilex–a database tool for integrating sign language corpus linguistics and sign language lexicography. In *LREC 2008 Workshop Proceedings. W 25: 3rd Workshop on the Representation and Processing of Sign Languages: Construction and Exploitation of Sign Language Corpora. Paris: ELRA*, pages 64–67.

Krause, T. and Zeldes, A. (2016). ANNIS3: A new architecture for generic corpus query and visualization. *Digital Scholarship in the Humanities*, 31(1):118–139.

Krause, T., Leser, U., and Lüdeling, A. (2016). graphANNIS: A Fast Query Engine for Deeply Annotated Linguistic Corpora. *JLCL*, 31(1):iii–25.

Lezius, W. (2002). TIGERSearch Ein Suchwerkzeug für Baumbanken. *Tagungsband zur Konvens*.

Odebrecht, C. (2012). Lexical Bundles. Eine korpuslinguistische Untersuchung. Master's thesis, Humboldt-Universität zu Berlin, Philosophische Fakultät II.

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

Schmidt, T. and Wörner, K. (2014). Exmaralda. In Ulrike Gut Jacques Durand et al., editors, *Handbook on Corpus Phonology*, pages 402–419. Oxford University Press.

Voormann, H. and Gut, U. (2008). Agile corpus creation. *Corpus Linguistics and Linguistic Theory*, 4(2):235–251.

Distributed Corpus Search

Radoslav Rábara, Pavel Rychlý, Ondřej Herman
Masaryk University, Faculty of Informatics
Botanická 68a, 602 00 Brno
{xrabara,pary,xherman1}@fi.muni.cz

Abstract

Available amount of linguistic data raises fast and so do the processing requIrements. The current trend is towards parallel and distributed systems, but corpus management systems have been slow to follow it. In this article, we describe the work in progress distributed corpus management system using a large cluster of commodity machines. The implementation is based on the Manatee corpus management system and written in the Go language. Currently, the implemented features are query evaluation, concordance building, concordance sorting and frequency distribution calculation. We evaluate the performance of the distributed system on a cluster of 65 commodity computers and compare it to the old implementation of Manatee. The performance increase for the distributed evaluation in the concordance creation task ranges from 2.4 to 69.2 compared to the old system, from 56 to 305 times for the concordance sorting task and from 27 to 614 for the frequency distribution calculation. The results show that the system scales very well.

Keywords: corpus, parallel, distributed, concordance

1. Introduction

Every year, the amount of text produced and stored increases, and so do the requirements to process and search it. Some of the largest corpora we build for use in Sketch Engine (Kilgarriff et al., 2014) now take months to compile, and their searching leaves a lot to be desired even on today's state-of-art machines, mainly due to lack of parallelization and storage bottlenecks.

Building on the approach described in (Rábara et al., 2017), where we report on our experiments in acceleration of corpus search using shared-memory multiprocessor machines, we developed an extension to the Manatee corpus management system (Rychlỳ, 2007) which allows us to distribute corpus operations over a cluster of commodity computers.

2. System description

The distributed system is based on the reimplementation of the Manatee corpus manager in Go.

We employ the MapReduce model, where machines in a cluster work on a local part of the data in isolation – *map* it to a partial result. These results are then propagated through the cluster and collated or *reduced* to obtain the final result. Our architecture uses multiple servers and a single client. The client schedules the work to be done, manages the servers, and handles interactions with the user. The servers are where the data is stored and where the performance intensive processing is carried out. The machines within the cluster communicate using a custom HTTP based protocol. The requests transmitted to the servers are encoded as JSON, while the much larger results are returned in a denser and more performant Protocol Buffers based format, as the encoding and decoding steps turned out to be a noticeable performance bottleneck.

2.1. Implemented features

Currently, only the most fundamental, but at the same time most difficult to implement, operations have been implemented. These are query evaluation, concordance building, sorting and frequency distribution calculations.

Corpus compilation is done in the same way as in the current Manatee system, except that each of the servers compiles its own local part of the corpus. Currently, we do not build any distributed indices and each part of the distributed corpus is a standard Manatee compatible corpus on its own. Uploading of the corpora to the servers is done out-of-band without any intervention of the system itself by standard UNIX tools.

Some important functionality has not been implemented yet, such as fault tolerance and fail-over, as these were deemed unnecessary in a proof-of concept system. Implementing some functions, such as the Word Sketch, should be straightforward, while other functions, such as thesaurus, might end up being calculated with the help of the servers, but ultimately stored on and queried by the client itself.

2.1.1. Concordance building

Evaluating Corpus Query Language queries and building the resulting concordance is done similarly as in the original implementation. In the original implementation, concordances are stored as lists of positions in the corpus. The textual form is generated on the fly when user requests a specific part of the concordance. Similarly, sorting and filtering operations manipulate numerical representations of tokens. This is not possible in the distributed implementation, as the other workers have no knowledge of lexicons and contents of the other parts of the corpus. Therefore, we build the textual representation of concordance, including the contexts, immediately on the servers. The final result is obtained by concatenating the partial results on the client.

2.1.2. Concordance sorting

It is often required for the concordance to be sorted by some criteria. We handle this case by distributed merge-sort. Partial results are generated and sorted on the servers and then streamed the client where the last merging pass is carried out. An optimization which avoids transferring all the partial results is in place for the case where a specific page of the concordance is requested. We build a sort index on each

Query	Result size	C++ implementation		Go implementation	
		asynchronous	synchronous	cluster	single machine
`[word="work.*ing"]`	3,696,606	14.62	20.04	0.99	37.19
`[word="confus.*"]`	702,436	14.88	18.24	0.29	34.89
`[word="(?i)confus.*"]`	731,452	25.44	34.51	0.76	43.63
`[lc=".*ing" & tag="VVG"]`	231,346,778	61.10	242.51	5.01	222.76
`[lemma_lc="good"] [lc="plan"]`	20,804	6.18	6.27	0.46	18.94
`[word=".*ing"]`	371,767,766	240.00	626.94	4.18	241.77
`[tag="JJ"] [lemma="plan"]`	553,724	3.18	18.21	1.33	15.74
`"some" [tag="NN"]`	5,107,984	3.28	36.14	1.46	22.72
`[lc=".*ing" & tag!="VVG"]`	141,174,215	61.75	229.08	5.84	281.31
`[tag="DT"][lc=".*ly"] [lc=".*ing"] [word="[A-Z].*"]`	54,957	334.01	more than 3600	32.88	more than 3600
`[tag="DT"][lc=".*ly"] [lc=".*ing"] [word="[A-Z].*" & tag!="P.*"]`	29,053	344.57	more than 3600	35.44	more than 3600

Table 1: Concordance building performance

Query	C++ implementation	Go implementation	
		cluster	single machine
`[word="Gauss"]`	26.89	0.48	26.85
`[word="recurrence"]`	180.16	1.09	52.00
`[word="enjoyment"]`	410.08	1.35	123.93
`[word="test"]`	492.79	3.29	158.38
`[word="said"]`	266.69	4.51	100.77
`[word="a"]`	more than 3600 s	23.99	more than 3600 s
`[word="the"]`	more than 3600 s	54.73	more than 3600 s

Table 2: Concordance sorting performance

Query	Result size	C++ implementation	Go implementation	
			cluster	single machine
`[word="Gauss"]`	497	17.01	0.36	12.80
`[word="recurrence"]`	1,580	159.32	0.33	31.90
`[word="enjoyment"]`	4,841	361.91	0.59	101.56
`[word="test"]`	33,100	482.94	3.67	138.39
`[word="said"]`	208,676	147.50	5.29	67.25
`[word="a"]`	1,700,427	576.39	15.42	136.90
`[word="the"]`	3,716,817	1273.01	28.86	621.96

Table 3: Frequency distribution performance

of the servers, which speeds up random accesses to the local sorted concordance. The client then merges the partial sort indices to obtain a global index, which can be queried to obtain the location of the requested concordance lines within the cluster. As the concordance is already in a textual form, it is not necessary to perform additional lookups of the sorting keys in the corpus. The trade-off is larger amount of data that needs to be transferred between the workers. The merging step on the client is the bottleneck of this operation. The performance is strongly affected by the transfer and decoding of the data coming from the clients, so we only retrieve the pages of the partial result necessary to display the current final result page. To speed up the decoding, we chose a dense binary-based representation built on the Protocol Buffers toolkit.

2.1.3. Frequency distribution

The frequency distribution feature is used to generate various kinds of histograms or contingency tables, such as the counts of context words which appear at a particular position with respect to a CQL query. This is a demanding operation, as it might be necessary to transfer lot of data between the server and the client. Multiple context features can be specified and the number of values of each of them can be as large as the size of the lexicon, therefore the result size can grow exponentially as additional context features are added. The frequency distribution can be obtained in shorter time with the distributed architecture, but the maximum allowable size is still limited by the available memory of the client. The partial histograms are calculated and sorted on every server and then transferred to the client, where they are combined to obtain the complete distribution. To speed up the

3. Performance evaluation

3.1. Hardware and software environment

The tests were carried out on a cluster consisting of 65 diverse computers. Each of them configured with 16 GB RAM and Ivy Bridge or Haswell Intel Core i5 4-core processor, depending on the particular machine. These machines are standard desktop computers accessible by students for general use in computer labs, so we cannot claim that the environment is free from external influences, but we found that the results during off-hours are consistent and repeatable. The operating system used was Fedora Linux.

3.2. Evaluation corpus

The corpus we used to evaluate the system is enTenTen12 from our TenTen family of Web corpora (Jakubíček et al., 2013). It is an English corpus consisting of approximately 13 billion tokens. Each of the 65 machines processed a part of the corpus with 200 million tokens stored on its local hard-disk.

3.3. Results

3.3.1. Concordance building

The Table 1 shows the time in seconds which was necessary to compute the concordance for a few selected queries from our test suite. Each of the concordance lines contains the text representation of the keyword and at most 40 characters to the left and at most 40 characters to the right, including paragraph boundaries and a document identificator.

This benchmark tests the raw speed of the mostly sequential index reads and the construction of the textual concordance representation, which utilizes the lexicons and the attribute text. These two operations are seek-intensive, but well-behaved with respect to the caching behavior.

We measured the performance of the current C++ implementation in two modes. In asynchronous mode, the time needed for the retrieval of the first 20 rows is given. This represents the time necessary to serve the first page of results to the user and approximates the latency of the system.

In synchronous mode, we waited until the whole text representation of the concordance had been constructed.

The asynchronous mode is not significantly faster compared to the synchronous mode, relative to the amount of result rows. This is because after the first 20 lines have been processed, lexicons and large parts of attribute texts are cached in system memory, so generating the rest of the result is much less expensive.

Performance of the distributed implementation was measured on the 65 worker cluster in one case and on a single worker in the second case. Asynchronous query evaluation has not been implemented in this system – the results need to be combined from the different workers as their order and location on the servers is not known beforehand, but we would still like to preserve their order in the face of indeterminism. The speedup from the distributed implementation varies from 2.4 to 69.2 when evaluated over the whole test suite.

The performance difference between the new implementation and the current implementation on a single machine is caused by optimizing the new implementation for complex queries with large result sizes, on the order of 5 % of the whole corpus. These issues have been largely eliminated in subsequent versions of the software, but we didn't have the opportunity to reevaluate the performance on the cluster yet.

On a per-processor basis, the distributed system is less efficient by design, as the lexicons present on each of the servers have significant overlap, but the total amount of memory that can be used to cache the data is much larger. As the working sets are smaller, every processor can be utilized better, as there are less cache spills.

3.3.2. Concordance sort

The Table 2 shows the time in seconds necessary to evaluate a query, sort the result and generate a textual representation of a concordance for a given query. The concordance was sorted using the three lowercased words to the right of the keyword as the key. As the performance of query evaluation has been examined in the previous section, the queries chosen for the evaluation consist of a single word to match each, as only the bounds of every match are used in the subsequent step. The structure of the query itself matters little. so We chose the words by their frequency, which is listed in the Table 4, to account for the various possible result sizes.

Query	Frequency
`[word="Gauss"]`	2,132
`[word="recurrence"]`	28,927
`[word="enjoyment"]`	157,287
`[word="test"]`	1,625,427
`[word="said"]`	10,842,497
`[word="a"]`	241,926,311
`[word="the"]`	547,226,436

Table 4: Query result sizes

The speedup ranges from 56 to 305 compared to the current implementation and from 1.0 to 3.5 when running on a single machine, excluding the operations which took too long to complete. The result for the query `[word="Gauss"]` takes almost the same time for the current and the new implementation, even though the new implementation uses multiple cores on every machine. This is because of the low-level inefficiencies of the new implementation. Compared to the concordance building benchmark, the new implementation is able to catch up because the sorting step can be more easily parallelized.

3.3.3. Frequency distribution

The Table 3 shows the time in seconds necessary to evaluate a query and generate the frequency distribution with respect to a single token immediately to the right.

The speedup between the current and new implementations ranges from 27 to 614 and from 1.4 to 5.0 when a single machine is used. The performance increases for queries with large result sets are limited by the large amount of data which needs to be transferred from the servers to the client and by the merging step carried out on it. Inserting another layer of workers to help with partial result merging between the servers and the client could help to decrease the total time necessary, at the cost of a possible increase in latency. However, we would like to avoid a stratified architecture, as it introduces large amounts of complexity and points of failure into the system.

The design of the system predates the availability of abundant cheap memory, so it behaves well under memory pressure. When only query evaluation is considered, the system is not sensitive to memory pressure – indices are processed in streaming fashion from disk, and memory is only used for caching. This extends to the distributed implementation. Frequency distribution, on the other hand, is inherently memory intensive, and using the current approach it is necessary to store the whole result on the client, which therefore needs to have the same amount of memory installed as if it were the singular machine used in the C++ implementation. In the future, it might be beneficial to split the frequency distribution functionality into more pragmatic features tailored to the specific use cases instead of the generic and elegant approach used now.

4. Conclusion

We describe the architecture of a proof-of-concept distributed corpus management system we developed and evaluated its performance on a large corpus. The results show that the performance of the distributed system scales very well and can support large scale corpus processing without the need for fancy storage arrays and distributed filesystems. Some queries that cannot currently be feasibly evaluated in interactive scenarios are now within reach, allowing more detailed analyses.

Acknowledgements

This work has been partly supported by the Ministry of Education of CR within the LINDAT-Clarin project LM2015071 and by the Grant Agency of CR within the project 18-23891S.

Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlỳ, P., and Suchomel, V. (2013). The tenten corpus family. In *7th International Corpus Linguistics Conference CL*, pages 125–127.

Kilgarriff, A., Baisa, V., Bušta, J., Jakubíček, M., Kovář, V., Michelfeit, J., Rychlỳ, P., and Suchomel, V. (2014). The sketch engine: ten years on. *Lexicography*, 1(1):7–36.

Rábara, R., Rychlý, P., Herman, O., and Jakubíček, M. (2017). Accelerating corpus search using multiple cores. In Piotr Bański, et al., editors, *Proceedings of the Workshop on Challenges in the Management of Large Corpora and Big Data and Natural Language Processing (CMLC-5+BigNLP) 2017 including the papers from the Web-as-Corpus (WAC-XI) guest section*, pages 30–34, Mannheim. Institut für Deutsche Sprache.

Rychlỳ, P. (2007). Manatee/bonito-a modular corpus manager. In *1st Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 65–70.

Using Elasticsearch for Linguistic Analysis of Tweets in Time and Space

Adrien Barbaresi[◦], **Antonio Ruiz Tinoco**[•]
[◦] Academy Corpora, Austrian Academy of Sciences
Sonnenfelsgasse 19, 1010 Vienna
adrien.barbaresi@oeaw.ac.at
[•] Faculty of Foreign Studies, Sophia University
7-1 Kioi-cho, Chiyoda-ku, Tokyo, 102-8554
a-ruiz@sophia.ac.jp

Abstract

The collection and analysis of microtexts is both straightforward from a computational viewpoint and complex in a scientific perspective, they often feature non-standard data and are accompanied by a profusion of metadata. We address corpus construction and visualization issues in order to study spontaneous speech and variation through short messages. To this end, we introduce an experimental setting based on a generic NoSQL database (Elasticsearch) and its front-end (Kibana). We focus on Spanish and German and present concrete examples of faceted searches on short messages coming from the Twitter platform. The results are discussed with a particular emphasis on the impact of querying and visualization techniques first for longitudinal studies in the course of time and second for results aggregated in a spatial perspective.

Keywords: Social media, Twitter, NoSQL databases, Time series, Spatial data mining

1. Introduction

Web documents in general and computer-mediated communication in particular put linguists in front of new challenges. As empirical evidence is now widely used in linguistics, the Web turns to a major source, which changes the way research is conducted. That is why Leech (2006) speaks of linguists as "inhabiting an expanding universe". The shift towards web texts took place around the year 2000, while focus on computer-mediated communication (CMC) is closely related to the emergence of social networks in the course of the 2000s. The sharp decrease in publication of work documenting web corpus construction and use after 2008 hints at a further development towards a particular focus on CMC data, which can be explained by the growing popularity of short message services (also called microblogs) in the form of social networks. The latter often provide larger number of identifiable users as well as clear data on network activity and range, for example by tracking followers. Because of the profusion of data and metadata, schemes and methods are needed to live up to the potential of these sources. It is indeed widely acknowledged that social and humanities scientists need to acquire and develop the skills to do data analysis and experiment with the visualization tools necessary to manage and interpret big data (Manovich, 2011).

At the time of Web 2.0 and APIs, a URL is merely an access point, the resulting website is often tailored not only to what the user has just typed in or said, but to a whole navigation and engagement history. Furthermore, the actual text may become accessory compared to the profusion of metadata which encase it. The clearly structured machine-readable formats combine up to hundreds of different fields ranging from previously studied information such as the user name or the date of posting to new, typical information such as the number of followers or the time zone of the user. That being said, not all information is as relevant or objective as one would expect, so that it has to be refined before reaching any conclusion.

Following from a first approach to download and indexing (Barbaresi, 2016), the present article tries to document both research methodology and research objects in an articulated fashion. Beyond the selection of a medium and the collection and preprocessing of messages, our work follows from two distinct procedures for handling social media information (Tsou and Leitner, 2013): "transform the original information formats into analytic forms" and "explore multiple analytical methods", to try to answer the following research question: "What should a comprehensive space-time analysis framework look like for different scenarios or questions?". Consequently, the visualizations introduced here are an attempt to find "the best combination of analytical methods to analyze social media information in depth from both temporal and spatial aspects" (Tsou and Leitner, 2013). We address corpus construction and visualization issues in order to study spontaneous speech and variation through short messages. We present and discuss an experimental setting to observe language through tweets, with a particular emphasis on the impact of visualization techniques on time series (messages seen in the course of time) and space (aggregated projections on maps of geolocated messages).

2. Experimental setting

2.1. Twitter as a source

The interest in Twitter is generally considered to reside in the immediacy of the information presented, the volume and variability of the data contained, and the presence of geolocated messages (Krishnamurthy et al., 2008). Other social networks do not deliver the same amount of text, especially for languages other than English, for example on Reddit (Barbaresi, 2015). Most importantly, they cannot be deemed as stable in time in terms of popularity and API access (Barbaresi, 2013). Nevertheless, because of access restrictions – mostly mechanical constraints on

the free streaming access point – it is not possible to retrieve all tweets one would need. It is indeed necessary to enter search terms or a geographic window, which may greatly affect results especially for highly frequent keywords (Ljubešic et al., 2014). The API is supposed to deliver a random sample representing a volume of about 1% of all tweets when used with a worldwide geographic window.

Since August 2009, Twitter has allowed tweets to include geographic metadata, which are considered to be a valuable source for performing linguistic studies with a high level of granularity, although the geolocation is not always accurate. Currently, the public Twitter Application Programming Interfaces (APIs) can provide five types of geocoding sources: geo-tagged coordinates, place check-in location (by way of a bounding box), user profile location, time zones, and texts containing explicit or implicit locational information (Tsou et al., 2017). The geo-tagged coordinates are the most frequently used type of information. The geolocalized messages can conveniently be projected on maps, which is highly relevant for various research fields, for instance variation studies in linguistics. That being said, it is important to note that geolocated tweets are a small minority, with estimates as low as 2% of all tweets (Leetaru et al., 2013). However, even the bounding boxes used to retrieve tweets do not always function as expected due to systematic errors (Tsou et al., 2017). Additionally, the geolocation results of profile locations are not a useful proxy for device locations, and the success at being able to place users within a geographic region varies with the peculiarities of the region (Graham et al., 2014).

From the point of view of corpus and computational linguistics, tweets are both highly relevant and difficult to process. Short messages published on social networks constitute a "frontier" area due to their dissimilarity with existing corpora (Lui and Baldwin, 2014), most notably with reference corpora. Some metadata are more useful than others, and some languages fit more easily into the allocated space than others (previously 140 characters, now 280 for most languages). Regarding the content itself, the quantity of information in general or the relevance for linguistic studies in particular may vary greatly. In spite of the restrictions on the API, the delivered volume may already be sufficient for diverse types of studies, and focusing on a given geographical region can be a way to provide enough relevant linguistic evidence. After appropriate filtering and selection, it is possible to draw maps to compare the geolocations of tweets with population density as a preliminary to study language variation (Arshi Saloot et al., 2016) or to use as input for classification tasks to determine language use in terms of variants on the social network (Alshutayri and Atwell, 2017).

2.2. Corpus building

While some studies ground on a collection process which is limited in time, the corpora described in this article are monitor corpora, as data collection goes on they grow with time. So-called "heavy tweeters" (Krishnamurthy et al., 2008) as well as peculiarities of the API (Morstatter et al., 2013) raise the question of sampling processes. The random sampling methodology used by Twitter to generate the streams of tweets is rarely put into question. This means that steps have to be taken in order to minimize or at least to assess the impact of differences in user activity as well as potentially unknown sampling biases. Studies have shown that it is desirable to gather a set of users which is both large and diverse (Zafar et al., 2015), so that the collection process is opportunistic. Such steps are described in previous work. It is possible to take decisions based on relevant metadata such as the number of followers or retweets as well as on information contained in the tweets themselves, such as the mention "RT" for retweet (Ruiz Tinoco, 2013). Additionally, it is possible only to take tweets coming from selected sources into account, for example by review the source fields in the collected tweets and focusing on common access points and clients (Tsou et al., 2017), most notably the website itself and the official Twitter mobile apps.

2.3. A suitable database infrastructure

The volume of storage space required to record and process the tweets is on the order of magnitude of several terabytes. Web data are a typical challenge for linguists working at public research institutions who do not dispose of large amounts of computing power (Tanguy, 2013). Additionally, Twitter data come in a form which has to be refined to suit the needs of linguists, as not all information and all metadata fields are linguistically relevant. In order to keep up with the challenges related to data structure and growing amount of tweets, we present our search engine of choice. The interest of NoSQL databases is known as regards the feature-rich content returned by the Twitter API (Kumar et al., 2014), they make it possible to access the corpus and see through it in various ways by using faceted searches. Their logic also supports indexing a variable number of metadata and efficiently divide the corpus into several subcorpora. In that sense, our purpose is to be opportunistic enough during corpus creation in order to allow for subcorpora which match particular interests.

Two main components of the open-source ELK stack (Elasticsearch, Logstash, Kibana) are used, namely Elasticsearch[1] to index the tweets and its front-end Kibana[2] to provide a user-friendly interface to queries, results, and visualizations. Installation and configuration are straightforward on most platforms, it is also possible to directly append servers to an existing cluster. Additionally, a sharding structure is implemented: shards (whether on the same machine or not) can be activated or deactivated to suit particular needs. Even with a single-server installation, it is convenient to process large amounts of tweets, that is on the basis of 10 Gb of data collected per day on the streaming API. The creation of subcorpora is possible through facets corresponding to a number of constraints acting on the text or the metadata (countries, precise time or date intervals, geographical window, etc.). Finally, the software is open-source and currently updated frequently, which gives access

[1] https://www.elastic.co/ Elasticsearch seems to be among the top-10 most popular database software at the moment https://db-engines.com/en/ranking

[2] https://www.elastic.co/de/products/kibana

to the latest optimizations or customizations (for example through Kibana's plugins).

Although it is not primarily a search engine for linguists, Elasticsearch takes advantage of the native JSON format of the tweets as well as of a number of relevant field types after a subsequent mapping, which allows for refined queries on text and metadata, which in turn ca be relevant for linguists, as we discuss in the remainder of this article. In order to give a user-friendly access to the results, dashboards can be configured out of a series of indicators. Despite its advantages for the structuration of non-standard data, the main drawback of nested JSON format resides in the lack of familiarity or compatibility with the formats commonly used in corpus linguistics, however the integration is possible through a number of conversion tools (e.g. plugins). The main drawbacks result at the time being from the built-in linguistic processing as well as a lack of integrated linguistic annotation. Considering non-standard speech, the standard lemmatization/normalization of queries and results by the search engine may be imprecise. Language-specific analysis modules can be selected, the default lemmatization is employed here due to the multilingual nature of our data, so that tokens can mainly be accessed on token or surface level.

3. Faceted querying and visualizations

In this section, we present three case studies focused on structured data synthesis in order to demonstrate the characteristics of language use on Twitter as well as the kind of information made available by our experimental setting.

3.1. Results presented as a dashboard

The first case deals with the amount of information available, which has to be filtered and presented it a synthetic fashion in order to allow for linguistic interpretation. We start from a classical display of results in corpus linguistics, the "word in context" (or KWIC) feature. Figure 1 shows a version of it which has been adapted to the tweets: results from several fields are aggregated into a dashboard, in that case the date, the text of the tweet (with the actual results), the name as chosen by the user, the "screen name" or account ID on Twitter, and the follower count. That way, it is straightforward to make a number of assumptions regarding the status of both user and tweet (for example concerning their influence). This example concerns colloquial German, where the contraction of *denkst du* in *denkste* has been considered to be typical for computer-based communication (Bartz et al., 2013). The query[3] contains the search for the exact expression *denkste* (without normalization), it also restricts the context to tweets in German which are not explicit retweets.

3.2. Combined time series

In a second example, we use the leverage provided by the amount of data to shed light on particular phenomena and use and visualization to corroborate hypotheses on language. Since metadata include the time of posting, it is possible to split the corpus in units of time. It is also quite natural to look at the axis of time in the monitor corpus, both in a linear and in a aggregated way. We present a longitudinal study which would be costly and tedious to realize in a classical fashion but where comparable information can be derived from the tweets themselves without artefacts linked to metadata. Figures 2a and 2b display the results of a time series query comparing two characterizing variants in Spanish in the course of the day: *buenos días* and *buen día*, Figure 2a focuses on tweets sent from Argentina while Figure 2b focuses on tweets sent from Spain, both are rendered using the Kibana's timelion plugin.[4] While a global figure would show patterns relative to the time zone of the users, these two uncouple the information to show the difference: a predominancy of *buen día* in Argentina and of *buenos días* in Spain, with a roughly similar pattern in the course of time highlighting that this expression is almost exclusively used in the morning. All this information is gathered in a suitable fashion for variation studies and is interpretable provided it is presented with the adequate circumspection.

3.3. Spatial analysis

Finally, our collection processes and infrastructure allow for the spatial studies on languages. A higher granularity of both queries and display is possible, as well as a direct access to the geolocalized tweets, which are then naturally interpretable in the form of a map. The tweets are automatically grouped by built-in clustering processes. The circles on the map display the number of tweets available for a particular place or region.

In order to illustrate the immediacy and practicality of the information available this way, we take Spanish diminutives as example, as there is a fair proportion of geolocated tweets to be found from different countries. There are several known diminutives for the word *café*, Figure 3a depicts the spatial distribution of tweets for the token *cafecito*, mostly in Central and South America, whereas Figure 3b focuses on *cafelito*, nearly exclusively to be found on the Spanish Peninsula. This comparison on maps using millions of tweets collected in 2017 confirms empirically this fact known to variation studies. On this order of magnitude, map processing from already indexed data is a matter of seconds and thus suitable for exploratory research from a linguistic and from a practical point of view.

3.4. Discussion

Twitter is a particular medium from an informational as well as from a sociological point of view. Age or gender biases are difficult to assess (Peersman et al., 2011), although they are certainly impacting both the structuration of the networks and the linguistic analysis. Additionally, a corpus may be affected by certain trends or keywords, beyond simple retweets or more complex repetitions of certain patterns. "Early adopters" and "power users" can distort a corpus quantitatively and qualitatively on the side of the observer and at the same time influence other users in their use of the language on the productive side.

Additionally, our experimental setting is not neutral and greatly contributes to shape the potential experiments. Nu-

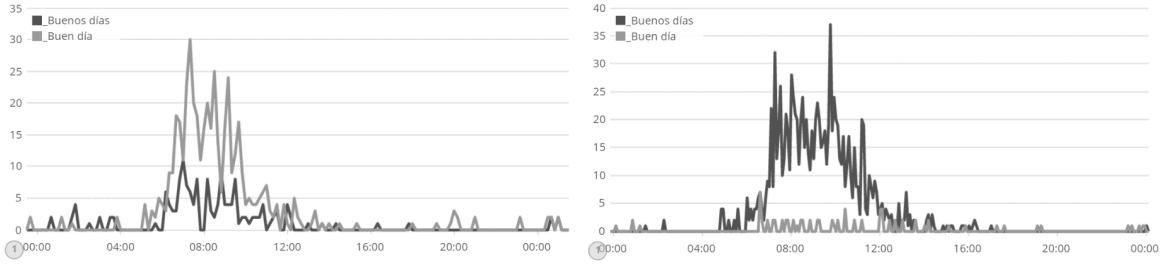
[3] text:"denkste" AND lang:de AND retweeted:False

[4] https://www.elastic.co/guide/en/kibana/current/timelion.html

text:denkste AND lang:de AND retweeted:false

@timestamp	text	user.name	user.screen_name	user.followers_count
April 16th 2017, 04:32:44.000	watt denkste @SirQLate ? Bombe, oder? https://t.co/V86HNpnphN	Frithjof Klepp	yakkul	89
March 8th 2017, 20:40:31.000	Da denkste oh lecker frisches Basilikum in der Tomatensoße und dann ist es Koriander...	Nadja	Suki9911	52
April 23rd 2017, 20:08:46.000	Da denkste hast nen schönen Sonntag, sind die Fellnasen andere Meinung https://t.co/rbd0CtuHbF	Papa Beck	_Papa_baer_	815
April 30th 2017, 03:00:00.000	achte isn typischer fall von denkste! bäm bäm bäm bäm bäm bäm bäm bäm	Rathausuhr Neukölln	rh_neukoelln	1,421
April 17th 2017, 16:40:20.000	Präsid.-kandidat #Macron: Deutschlands wirtsch. Stärke nicht mehr tragbar. Sollen wir schlechter werden? Denkste! https://t.co/YPpkG8C3wT	Heinz Scholz	HSderSchreiber	239
April 9th 2017, 01:22:25.000	Und dann fühlen die Finger zärtlich über die Veltins Relief-Flasche, und zack, denkste an Rudi. #S04	Torsten Wieland	TorstenWieland	4,507
April 29th 2017, 17:26:33.000	"130kmh sonne Rotze da denkste de stehst auf der kackautbahn" - mein Vater, 54, hat sich wiedermal blitzen lassen	Johann Scholz	ScholziWins	64
April 24th 2017, 05:20:00.000	Scheisse Da denkste die ganze zeit "Geile TL Heute" zack biste auf der @AnnaNymchen ihre Seite. Gönnt euch	Schattendasein	140zeichensucks	215
May 2nd 2017, 07:49:48.000	Da denkste, irgendwas in der Ecke bewegt sich doch und dann ist es Tim der auf dem Balkondeckenstapel hinter dem Vo… https://t.co/PgzP43ciqj	Çapulcu B	_blickwinkel_	3,728
April 5th 2017, 18:26:01.000	Sitz im Garten und hau eben mal ne Salami weg. Dann guckste auf die Inhaltsstoffe und denkste tut das not der ganze… https://t.co/yIypxnSNTW	↔inFarbeundBunt ↔	sabinehartl1977	382

Figure 1: Detail/qualitative analysis in the dashboard mode, exact search for the contraction *denkste* in tweets identified as being in German and excluding retweets



(a) Focus on Argentina

(b) Focus on Spain

Figure 2: Two variants of salutation (singular and plural), visualized through Kibana according to the hours of use (in each respective time zone)

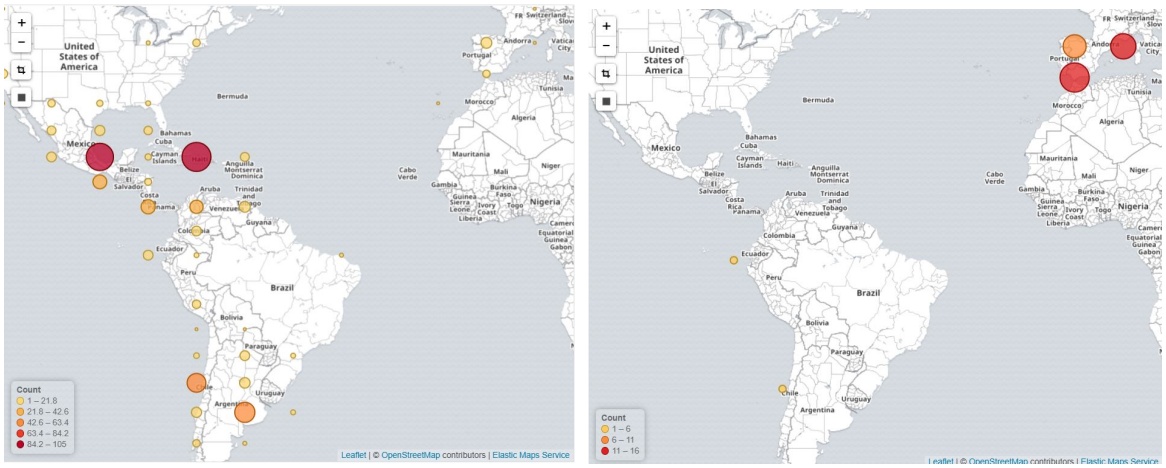
meric indicators tend to be favored as they allow for immediate realization of graphs such as bar charts. The analysis of text can get much more complicated, starting from the linguistic annotation tasks: language identification of tweets is error-prone. Catalan for example very often resorts to the "undetermined" category according to Twitter data. Furthermore, tweets are used for various purposes and may entail a diverging amount of "text" in a traditional linguistic sense (statements, replies, phatic formulas as opposed to hashtags, gifs, emoticons, and emojis).

Installations and tweet corpora are maintained at both institutions[5], they are used by colleagues and students alike, mostly for exploratory studies on spontaneous non-standard language. In this context, despite the lack of deep linguistic analysis, the interface is more user-friendly, easier to explain to students for example, and thus more directly usable for linguistic studies. The aggregation of data into a dashboard provides a way to find the right balance between profusion of data and relevance. We can still highlight two main artefacts of the apparatus in this case. First, there is a strong tendency to adjust the queries to the output, that is to test for hypotheses which lead to clear-cut results. Second, the projection on maps is the most prominent feature. Geolocated tweets distinguish this platform from other social networks, and corpus users are fond of reactive, interpretable maps.

4. Conclusion

The actual contents of a web corpus can often only be listed with certainty once the corpus is complete. In fact, corresponding to the potential lack of information concerning

[5]Academy Corpora, Austrian Academy of Sciences and Faculty of Foreign Studies, Sophia University.



(a) cafecito (b) cafelito

Figure 3: Spatial distribution of two diminutives of the word *café* in tweets

the metadata of the texts is a lack of information regarding the content, which has to be recorded and evaluated a posteriori, in a *post hoc* evaluation (Baroni et al., 2009). The notion of *a posteriori* analysis is a key concept regarding the study of tweets, whereas corpus design as considered by the tradition of corpus linguistics is systematically *a priori*. This bears both a risk and a chance: the linguistic relevance of the documents included is harder to assess, but it is also possible to determine new structural elements and discover relations in the data, for instance linguistic phenomena.

In this sense, we addressed corpus construction and visualization issues in order to study spontaneous speech and variation through short messages, notably metadata such as a location embedded in tweets. The technological stack based on Elasticsearch and Kibana has convinced us by its stability, scalability, and ease of use, although it ought to be complemented by further refinements and annotations in order to suit the needs of linguists.

We believe that our experimental setting can bring linguistic studies closer to actual language use. To this end, the adequation of the corpus with a given research goal has to be assessed. It is perfectly possible to adapt the geometry of the corpus to target a particular user type, region, or language. Yet beyond the scope of geographic variation as traditionally seen by examining utterances in a lexical or syntactic or other linguistic aspects, the study of online social networks also opens up new possibilities regarding user involvement and activity or general characteristics of the populations, features which would have needed a particular data collection effort in the past and which were not put into focus in variation studies.

5. Bibliographical References

Alshutayri, A. and Atwell, E. (2017). Exploring Twitter as a Source of an Arabic Dialect Corpus. *International Journal of Computational Linguistics (IJCL)*, 8(2):37–44.

Arshi Saloot, M., Idris, N., Aw, A., and Thorleuchter, D. (2016). Twitter corpus creation: The case of a Malay Chat-style-text Corpus (MCC). *Digital Scholarship in the Humanities*, 31(2):227–243.

Barbaresi, A. (2013). Crawling microblogging services to gather language-classified URLs. Workflow and case study. In *Proceedings of the 51th Annual Meeting of the ACL, Student Research Workshop*, pages 9–15.

Barbaresi, A. (2015). Collection, Description, and Visualization of the German Reddit Corpus. In *2nd Workshop on Natural Language Processing for Computer-Mediated Communication, GSCL conference*, pages 7–11.

Barbaresi, A. (2016). Collection and Indexing of Tweets with a Geographical Focus. In *Proceedings of CMLC-4, LREC 2016*, pages 24–27.

Baroni, M., Bernardini, S., Ferraresi, A., and Zanchetta, E. (2009). The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Bartz, T., Beißwenger, M., and Storrer, A. (2013). Optimierung des Stuttgart-Tübingen-Tagset für die linguistische Annotation von Korpora zur internetbasierten Kommunikation: Phänomene, Herausforderungen, Erweiterungsvorschläge. *JLCL*, 28(1):157–198.

Graham, M., Hale, S. A., and Gaffney, D. (2014). Where in the world are you? Geolocation and language identification in Twitter. *The Professional Geographer*, 66(4):568–578.

Krishnamurthy, B., Gill, P., and Arlitt, M. (2008). A Few Chirps about Twitter. In *Proceedings of the First Workshop on Online Social Networks*, pages 19–24. ACM.

Kumar, S., Morstatter, F., and Liu, H. (2014). *Twitter Data Analytics*. Springer.

Leech, G. (2006). New resources, or just better old ones? The Holy Grail of representativeness. *Language and Computers*, 59(1):133–149.

Leetaru, K., Wang, S., Cao, G., Padmanabhan, A., and Shook, E. (2013). Mapping the global Twitter heartbeat: The geography of Twitter. *First Monday*, 18(5).

Ljubešic, N., Fišer, D., and Erjavec, T. (2014). TweetCaT: a Tool for Building Twitter Corpora of Smaller Languages. *Proceedings of LREC*, pages 2279–2283.

Lui, M. and Baldwin, T. (2014). Accurate Language Identification of Twitter Messages. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)@ EACL*, pages 17–25.

Manovich, L. (2011). Trending: The promises and the challenges of big social data. *Debates in the digital humanities*, 2:460–475.

Morstatter, F., Pfeffer, J., Liu, H., and Carley, K. M. (2013). Is the Sample Good Enough? Comparing Data from Twitter's Streaming API with Twitter's Firehose. In *Proceedings of ICWSM*.

Peersman, C., Daelemans, W., and Van Vaerenbergh, L. (2011). Predicting age and gender in online social networks. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 37–44. ACM.

Ruiz Tinoco, A. (2013). Twitter como Corpus para Estudios de Geolingüística del Español. *Sophia Linguistica: working papers in linguistics*, 60:147–163.

Tanguy, L. (2013). La ruée linguistique vers le Web. *Texto! Textes et Cultures*, 18(4).

Tsou, M.-H. and Leitner, M. (2013). Visualization of social media: seeing a mirage or a message? *Cartography and Geographic Information Science*, 40(2):55–60.

Tsou, M.-H., Zhang, H., and Jung, C.-T. (2017). Identifying Data Noises, User Biases, and System Errors in Geo-tagged Twitter Messages (Tweets).

Zafar, M. B., Bhattacharya, P., Ganguly, N., Gummadi, K. P., and Ghosh, S. (2015). Sampling Content from Online Social Networks: Comparing Random vs. Expert Sampling of the Twitter Stream. *ACM Transactions on the Web (TWEB)*, 9(3):12.

How to Get the Computation Near the Data: Improving Data Accessibility to, and Reusability of Analysis Functions in Corpus Query Platforms

Marc Kupietz, Nils Diewald, Peter Fankhauser
Institut für Deutsche Sprache
R5, 6-13, D-68161 Mannheim
{kupietz, diewald, fankhauser}@ids-mannheim.de

Abstract

The paper discusses use cases and proposals to increase the flexibility and reusability of components for analysis and further processing of analysis results in corpus query platforms by providing standardized interfaces to access data at multiple levels.

Keywords: Corpus Query Platform, Interoperability, Reusability, Extensibility, Legal Issues

1. Introduction

Compared to other disciplines that deal with big research data, in linguistics it is on the one hand particularly important and on the other hand particularly difficult to equip research software with satisfactory functionalities for data analysis. It is important because the research data usually is not only too big to move, but also – due to IPR and license restrictions – often not allowed to move (Kupietz et al. 2010, 2014; Kamocki et al. 2016). For this reason, researchers who want to analyse corpora cannot simply download the data and apply their own software tools. Providing generally satisfactory functionalities in central corpus analysis platforms, however, has proven difficult. One reason for this is that the question, how language data should be analysed, depends very much on the research aims, and is itself a key research field with rapid development. As a consequence, provided functionalities will typically not cover all areas of interest and will be outdated quickly. Finding solutions for this dilemma is even more important as a quick and general availability of methods for analysis and for the visualization of analysis results, and accordingly a certain nudge towards methodological canonicalization is likely to foster scientific progress in disciplines that deal with language data.

In this paper we will review current approaches (including our own approach with KorAP) to interoperable language analysis and discuss their limitations. We then describe typical analysis use cases from which we derive the main data structures needed for their support. On this basis, we outline a basic API, and discuss the resulting data access requirements.

2. Previous work

2.1. CLARIN

A lot of valuable and groundbreaking work with respect to the reusability of language resources and technology has been done within the European Research Infrastructure for Language Resources and Technology CLARIN. Apart from corpus encoding standards, and best practices, particularly relevant for the subject of this paper are the experiences with the web-service orchestration and tool-chaining platform for corpus processing WebLicht (Hinrichs, Hinrichs and Zastrow 2010). WebLicht itself, however, is based on a distributed network of web-services between which data is transferred while analysis functions are immobile. Thus the WebLicht approach does not address license and IPR restriction problems, typical in corpus linguistics.

In general, the task addressed in this paper will complement the work done in CLARIN, in the sense that CLARIN work focuses on commonly used base infrastructures and existing resources and technologies, while the task here is focussed on a very specific class of use cases and applications.

2.2. KorAP

Increasing data reusability and accessibility for different applications using different methods was one of the key targets of KorAP, the corpus analysis platform developed at the IDS starting in 2011 (Kupietz et al. 2014). From the beginning of the KorAP project, it has been clear that the core developers at the IDS would not be able to develop and maintain all desired functionalities to satisfy all potential users.

At an early stage of the KorAP's design phase (Bański et al. 2012, p. 2906), the intended approach to solve the problem was to roughly follow Jim Gray's (2004) principle, *if the data cannot move put the computation near the data*, by providing a mobile code sandbox where users can run their own "KorAPp" code with controlled output in order to meet license restrictions (Kupietz et al. 2010). However, due to high expected development costs of such a sandbox, its maintenance and the required hardware, this approach was only pursued in a manual way: DeReKo samples were sent to interested users to let them adapt their software to the DeReKo encoding format, the software then was applied manually on IDS servers, and the controlled output was sent back to the users (Kupietz et al. 2014).

Proposed levels of access

To simplify this work as much as reasonable, since 2014 (Bański et al.) KorAP follows an alternative multi-level approach for making the data accessible to user code depending on the respective task, with the following levels of access:

- **Corpus**: Corpus-level access is typically most suitable for tasks that (a) require whole texts as input and/or (b) need to add new annotations to the corpus data. A typical example for such a task is topic domain or text type

classification. It can be carried out by means of (manually applied) mobile code as described above.

- **Multiple (Backend) Instances:** This level, or rather approach, is ideally applicable for tasks relying on multiple, physically separated corpora and where identical functionalities are expected as e. g. for contrastive studies on comparable corpora (Cosma et al. 2016, Kupietz et al. 2017). In addition, however, this approach can also be a complementary alternative to standardized data access. For example if a corpus query tool A is used, but functionalities of tool B are required, in some cases the easiest solution can be to convert the corpus data to the format required by tool B and run an instance of tool B in addition to tool A.
- **Web-API**: This level seems ideally applicable for tasks that require search results. In the case of KorAP, the interface is specified by the Kustvakt REST-API and the KoralQuery protocol (Bingel and Diewald 2015); optionally executed close to the data to avoid network latencies ("fast lane").
- **Open Source**: This level is ideal for tasks that require or suggest extensions to core functionalities. In the case of KorAP, such extensions can be proposed for review and submitted via Gerrit[1][2]. For functionalities for which alternatives are specifically welcomed, an interface definition (in the Java sense) could streamline this extension process.

Currently, the access at these different levels is, however, mostly KorAP-specific and not standardized. Thus, the proposed levels of data-access primarily serve to complement and to facilitate further steps of canonicalization and standardization.

2.3. Other corpus query tools

Probably all existing corpus query tools support mechanisms for exporting query results in different formats to allow further processing. The corpus workbench (Evert and Hardy 2011), for example, provides a tabulate command to aggregate query results into tables that can then be used for further statistical analysis.

A corpus query tool that goes very far with providing interfaces for data access is the CorpusExplorer[3] (Rüdiger 2018) which offers an extensible large variety of export formats and an SDK to allow users to develop their own functions for analysis and further processing.

Nederlab (Brugman et al. 2016) provides an R based visualization service (Komen 2015) as a separate service component to further process search results, especially for visualizations, and is meant to support even user provided custom R modules.

3. Use cases

This list of example use cases for exchangeable analysis modules is not intended to be complete. It rather reflects our own interests, the experiences with the DeReKo user community, and mainly serves as an overview of possible application classes and their requirements.

3.1. Collocation Analysis

An obvious and simple use case for interoperable analysis modules is collocation analysis (Sinclair 1991), which is offered in nearly every corpus platform and one of the most widely used and most often varied methodologies in corpus linguistics.

The standard case: cohesion of word pairs

In the standard case, collocation or co-occurrence analysis measures the association between co-occurring words. I. e. it assigns an association score to word pairs of the vocabulary observed in a corpus based on their respective total frequencies and their co-occurrence frequency within a certain context-window, e. g. [-5, 5] around the target word, so that high scores indicate strong attraction, low scores indicate weak attraction, and scores can be used to rank the word pairs according to the strength of their attraction (Evert 2009).

The results of collocation analysis can be used for finding recurrent syntagmatic patterns and, by means of comparing vectors of association scores (*collocation profiles*) for finding paradigmatic relationships between words (Perkuhn 2007). For both scenarios exchangeable analysis functions and interfaces for further processing or visualizing the results would be desirable.

Higher-order collocation analysis

In higher-order collocation analysis (Keibel and Belica 2007) the analysis is applied recursively by taking the cohesive pairs found in one step as a single node and using some form of the found concordances as the corpus in the subsequent step.

It is already more difficult to define a sufficiently general standardized API for this simple extension, however, it might be a good example of functions that are more easily standardizable on an API level by using callbacks.

3.2. Corpus Comparison

The general goal of corpus comparison (Kilgariff 2001) is to analyse language use depending on text-external variables, such as mode (oral vs. written), genre, register, discipline, or time, in order to understand the correlation between text-internal features with text-external variables. Analysis techniques comprise multivariate analysis (e. g. Biber 1993), cluster analysis, and classification (Teich and Fankhauser 2010) together with feature selection and ranking (Fankhauser et al. 2014, Teich et al. 2014).

In terms of data structures, the typical workflow for corpus comparison is as follows. For feature selection, a query on the corpus is used to select a virtual subcorpus and features of interest. The query result consists of sequences of words, or more generally sequences of features (such as part-of-speech *n*-grams), from which bags of words can be derived. Crucial for this kind of analysis is that the query results are contextualized with the text-external variables.

[1] KorAP's Gerrit: `https://korap.ids-mannheim.de/gerrit/`

[2] Gerrit Code Review in general: `https://www.gerritcodereview.com/`

[3] `https://notes.jan-oliver-ruediger.de/software/corpusexplorer-overview/`

3.3. Provenance of analysis results: Linking back to the concordances

A feature that is generally desirable across many if not all use cases is the possibility to link back from aggregated representations of query or analysis results to the corpus texts that were the basis of the analysis. A typical example would be to allow users to click on tokens displayed in map-visualizations of distributional-semantic neighbourhoods (Keibel and Belica 2007, Fankhauser and Kupietz 2017), or in frequency graphs (Kupietz et al. 2017b: 327) in order to execute a query that shows the corresponding or underlying concordances. Such a feature is highly desirable because in typical exploratory workflows (Tukey 1977) the results of further processing of corpus analyses do not have the function to thoroughly display observations but rather to illicit the abduction of new hypotheses that need to be verified on the basis of the observed data (Keibel and Kupietz 2009, Jockers 2013, Perkuhn and Kupietz 2018: 87-88).

4. Data modelling

As exemplified in the use cases above, the two main data structures for text analysis are sequences of words and bags of words.

Sequences of words maintain the actual order of words in context of their use up to a certain length. Thereby, sequences comprise concordances, word *n*-grams, and, when adorned with frequencies, *n*-gram language models.

Bags of words disregard the order of words, but often represent larger contexts, such as documents or entire subcorpora, by means of vectors over the entire vocabulary of a corpus.

Word co-occurrences constitute an interesting case in between. On the one hand, they can be modelled by means of bigram sequences w_1w_2, on the other hand, as bags of words indexed by w_1 or w_2.

Both, sequences of words and bags of words, should be equipped with a (not necessarily unique) context identifier, which allows to associate them with text external variables, such as metadata, about the document or the subcorpus. Likewise, the words themselves can be equipped with identifiers, in order to associate word or position specific annotations, such as lemma or part-of-speech, and more generally link back to the corpus text.

5. Interfaces

To make analysis modules in corpus query systems reusable, standardized interfaces are required. As all use cases focus on very large corpus data and the introduced perspective is on data that requires restricted access due to license and IPR restrictions, a scenario of standardized web-service APIs seems to be obvious, although interface definitions could be adapted for programming library interfaces as well. The interface definition can be separated in three parts: the request protocol, the response format, and, in case the analysis results should be represented by the corpus query system or processed further, the analysis format.

Request Protocol

The request protocol specifies endpoints and parameters for data access. Regarding the presented example use cases, these are at least

- **Query endpoint**: Requires corpus definition (or document identifier), query definition (or positional identifiers), context definition (in characters, words, annotations, etc.), metadata fields to retrieve, annotations in the match to retrieve
- **Statistical endpoint**: Calculation of, e.g., numbers of tokens or occurrences in a defined corpus
- **Grouping endpoint**: Grouping of matches with frequency information

The request protocol probably requires batch processing for large request sets and a paging/cursor mechanism for large result sets. Existing APIs to get inspiration from include OpenSearch[4], SRU[5], and PortableContacts[6].

Response Format

The response format represents the accessible corpus data in a machine readable format, preferable in JSON (or JSON-LD, following recommendations from ISO TC37 SC4 WG1-EP) or XML for further processing. Existing formats to get inspiration from include RSS, Atom, and ActivityStreams[7].

Analysis Format

The results of the presented analysis methods can be serialized as data tables, therefore they may use a CSV format for further processing, or a serialization to JSON or XML. Existing APIs to get inspiration from include the Web Language Model API[8]. For visual integration in user interfaces, the data may be passed as image data or presented in (sandboxed) iframes. Existing APIs to get inspiration from include OpenSocial[9].

6. Data Access Requirements and Current Implementations

The use cases introduced above require means of access to corpus data that are not necessarily in the focus of corpus query engines, which are typically optimized for structured queries (formulated in corpus query languages like CQP, Christ 1994) on structured textual data, rather than for providing data structures suitable for further analysis. In addition, they may provide functionalities to define subcorpora or restrict access to the underlying corpus, because full access is limited due to legal constraints. To meet these requirements, most corpus query engines either rely on uniformly layouted documents (e. g. searching plain XML files using XQuery or XPath) or indexed corpus data. As operations on non-indexed data can computationally be expensive for very large corpora, indexed representations are prefered for most use cases involving data analysis.

Recent developments in corpus query engines focus on inverted indices, as used by BlackLab[10], MTAS[11] or KorAP

[4] http://www.opensearch.org/
[5] https://www.loc.gov/standards/sru/
[6] http://portablecontacts.net/
[7] http://activitystrea.ms/
[8] https://azure.microsoft.com/de-de/services/cognitive-services/web-language-model/
[9] https://github.com/opensocial
[10] http://inl.github.io/BlackLab/
[11] https://meertensinstituut.github.io/mtas/

(Diewald and Margaretha 2016) - although alternative approaches have proven to be useful as well (e. g. relational database models in ANNIS, Zeldes et. al 2009; or graph databases in graphANNIS, Krause et al. 2016). While inverted indices perform well on the task of structural queries (by providing for fast retrieval of textual units using a dictionary of surface terms, annotations, or metadata fields), they are not well suited for fast retrieval of contextual information necessary for the required data modelling for data analyses. The match information of a search, retrieved from the postings lists of an inverted index contain, at minimum, a document identifier (that can be used to recreate context on the document level) and optionally positional information (that can be used to recreate context of occurrences on the token or character level). However, the recreation of contexts (for example to retrieve all meta information of a document or to generate snippets) normally requires post-processing steps, involving additional data structures (see Manning et al. 2008, p. 158). Implementations like Apache Lucene[12] (the inverted index architecture behind BlackLab, MTAS, and KorAP) can provide fast access to stored metadata fields as well as primary data, that can be, in conjunction with positional information, be used to recreate textual context for a match. These additional data representations however have the disadvantage of introducing redundancy. Instead of a raw primary data string, a stored field may contain an annotation enriched representation as well (cf. COSMAS II, Bodmer 1996), introducing even more redundancy. This is useful to return context including annotational markup (see the use case for corpus comparison) or to limit the context according to annotation boundaries, like sentences. More elaborate engines make use of an additional forward index (see for example BlackLab[13], or prototype versions of KorAP[14]) to provide fast access to the raw data by positional information.

Additional data access is required for the proposed use cases regarding corpus statistics, for example to retrieve the number of tokens in a specified subcorpus, or the number of occurrences of context tokens in a subcorpus. Some of these information can be precalculated and stored in document associated meta fields, but the corpus query engine needs to provide methods for performant data aggregation as well.

To make data analysis components exchangeable, corpus query engines will be required to not only provide fast search capabilities, but also grant fast access to all underlying corpus data (primary data, annotation data, and metadata), based on document and positional information, while still respecting IPR restrictions.

7. Preliminary Conclusions

Improving the interoperability and extensibility of analysis and further processing functions in different corpus query engines seems desirable and feasible. However, the development and maintenance costs for supporting more sophisticated applications via canonical APIs seem – given the extensive experiences within the CLARIN project, the heterogeneity of corpus query tools and even the limited scope of use cases given in this paper – quite high. We, thus, recommend to start the canonicalization with functions that are of strong common interest as well as easily convergable. To this end, we presented some widely used applications for corpus analysis, and identified the main data structures to support them.

As flanking measures, we recommend to follow and to extend a multi-level approach as sketched above, already at a not (fully) standardized stage. This means for example to support standard corpus encoding and annotation formats, so that corpora and corpus analysis tools can be added and exchanged with manageable efforts and to support the open source level by encouraging the extension of corpus query systems with new analysis functions on the part of external users and developers. As always, standardization makes sense upto a point where the total costs for implementing and maintaining the required generality exceeds the total costs of achieving the desired results at the relevant places and maintaining their required reproducibility without a standard. Instead of guessing, where this break-even point will be, it seems reasonable to start with some safe candidates while not neglecting other (promising) ones.

8. References

Bański, P., Fischer, P. M., Frick, E., Ketzan, E., Kupietz, M., Schnober, C., Schonefeld, O., Witt, A. (2012). The New IDS Corpus Analysis Platform: Challenges and Prospects. In: Calzolari, N. et al. (eds.): Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12). Istanbul, Turkey, May 2012. European Language Resources Association (ELRA), 2012: 2905-2911.

Bański, P., Diewald, N., Hanl, M., Kupietz, M. and A. Witt (2014). Access Control by Query Rewriting: the Case of KorAP. In: *Proceedings of the 9th conference on the Language Resources and Evaluation Conference* (LREC 2014), European Language Resources Association (ELRA), Reykjavik, Iceland, May 2014: 3817-3822.

Biber, D. (1993). The Multi-Dimensional Approach to Linguistic Analyses of Genre Variation: An Overview of Methodology and Findings, Computers and the Humanities, 26: 331-345.

Bingel, J. and Diewald, N. (2015). KoralQuery – a General Corpus Query Protocol. In: Proceedings of the Workshop on Innovative Corpus Query and Visualization Tools at NODALIDA 2015, Vilnius, Lithuania, May 11-13, pp. 1-5.

Bodmer, F. (1996). Aspekte der Abfragekomponente von COSMAS-II. LDV-INFO. Informationsschrift der Arbeitsstelle Linguistische Datenverarbeitung, 8:112–122.

Brugman, H., Reynaert, M., van der Sijs, N., van Stipriaan, R., Sang, E. T. K., and van den Bosch, A. (2016). Nederlab: Towards a Single Portal and Research Environment for Diachronic Dutch Text Corpora. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slovenia. pp.1277-1281.

Christ, O. (1994). A modular and flexible architecture for an integrated corpus query system. In Proceedings of COM-

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

[13] `http://inl.github.io/BlackLab/file-formats.html`

[14] See `https://github.com/KorAP/Krawfish-prototype`

PLEX'94, 3rd Conference on Computational Lexicography and text Research, Budapest, Hungary. pp. 23-32.

Cosma, R., Cristea, D., Kupietz, M., Tufiş, D., Witt, A. (2016). DRuKoLA – Towards Contrastive German-Romanian Research based on Comparable Corpora. In: Bański, P. et al. (eds.):4th Workshop on Challenges in the Management of Large Corpora. Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Portorož, Slowenien. Paris: European Language Resources Association (ELRA), 2016: 28-32.

Diewald, N. and Margaretha, E. (2016). [200E?]Krill: KorAP search and analysis engine. In: Journal for Language Technology and Computational Linguistics (JLCL), 31 (1). 63-80.

Evert, S. (2009). Corpora and collocations. In Lüdeling, A. and Kytö, M. (eds.), Corpus Linguistics. An International Handbook. Mouton de Gruyter, Berlin.

Evert, S. and Hardie, A. (2011). Twenty-first century Corpus Workbench: Updating a query architecture for the new millenium. In Proceedings of the Corpus Linguistics 2011 conference, University of Birmingham, UK.

Fankhauser, P., Knappen, J. and Teich, E. (2014). Exploring and Visualizing Variation in Language Resources. Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14). Reykjavik, Iceland

Fankhauser, P. and Kupietz, M. (2017). Visualizing Language Change in a Corpus of Contemporary German. In: Proceedings of the 9th International Corpus Linguistics Conference. Birmingham: University of Birmingham, 2017.

Gray, J. (2004). Distributed Computing Economics. In: Herbert A., Jones K.S. (eds) Computer Systems. Monographs in Computer Science. Springer, New York, NY

Hinrichs, E., Hinrichs, M, and Zastrow, T. (2010). WebLicht: Web-Based LRT Services for German. In: *Proceedings of the ACL 2010 System Demonstrations*. pp. 25–29.

Jockers, M. L. (2013). Macroanalysis: Digital Methods and Literary History. Champaign, IL: University of Illinois Press.

Kamocki, P., Kinder-Kurlanda, K., Kupietz, M. (2016). One Does Not Simply Share Data. Organisational and Technical Remedies to Legal Constraints in Research Data Sharing -- building bridges between Digital Humanities and the Social Sciences. In: Witt, A., Kamocki, P. (2016). When DH Meets Law: Problems, Solutions, Perspectives. In Digital Humanities 2016: Conference Abstracts. Jagiellonian University & Pedagogical University, Kraków, pp. 104-108.

Keibel, H. and Belica, C. (2007). CCDB: A Corpus-Linguistic Research and Development Workbench. In: Proceedings of the 4th Corpus Linguistics Conference (CL 2007). Birmingham: University of Birmingham.

Keibel, H. and Kupietz, M. (2009): Approaching grammar: Towards an empirical linguistic research programme. In: Minegishi, Makoto/Kawaguchi, Yuji (Eds.): Working Papers in Corpus-based Linguistics and Language Education, No. 3. Tokyo: Tokyo University of Foreign Studies (TUFS), 2009. pp. 61-76.

Kilgarriff, A. (2001). Comparing Corpora. International Journal of Corpus Linguistics, 6(1): 1-37.

Komen, E. R. (2015). An insider's guide to the Nederlab R visualization webserver. Technical report, Nederlab.

Krause, T., Leser, U., Lüdeling, A. (2016). graphANNIS: A Fast Query Engine for Deeply Annotated Linguistic Corpora. In: Journal for Language Technology and Computational Linguistics (JLCL), 31 (1). 1-25.

Kupietz, M., Belica, C., Keibel, H. and Witt, A. (2010). The German Reference Corpus DeReKo: A primordial sample for linguistic research. In: Calzolari, N. et al. (eds.): *Proceedings of LREC 2010*. 1848-1854.

Kupietz, M., Lüngen, H., Bański, P. and Belica, C. (2014). Maximizing the Potential of Very Large Corpora. In: Kupietz, M., Biber, H., Lüngen, H., Bański, P., Breiteneder, E., Mörth, K., Witt, A., Takhsha, J. (eds.): *Proceedings of the LREC-2014*-Workshop Challenges in the Management of Large Corpora (CMLC2). Reykjavik: ELRA, 1–6.

Kupietz, M., Witt, A., Bański, P., Tufiş, D., Cristea, D., Váradi, T. (2017). EuReCo – Joining Forces for a European Reference Corpus as a sustainable base for cross-linguistic research. In: Bański, P. et al. (eds.): Proceedings of the Workshop on Challenges in the Management of Large Corpora and Big Data and Natural Language Processing (CMLC-5+BigNLP) 2017. Birmingham, 24 July 2017. Mannheim: Institut für Deutsche Sprache, 2017: 15-19.

Kupietz, M., Diewald, N., Hanl, M., Margaretha, E. (2017b). Möglichkeiten der Erforschung grammatischer Variation mithilfe von KorAP. In: Konopka, M., Wöllstein, A. (eds.): Grammatische Variation. Empirische Zugänge und theoretische Modellierung. Berlin/Boston: de Gruyter.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval, Cambridge University Press.

Perkuhn, R. (2007). Systematic Exploration of Collocation Profiles. In: Proceedings of the 4th Corpus Linguistics Conference (CL 2007), Birmingham. University of Birmingham.

Perkuhn, R. and Kupietz, M. (2018). Visualisierung als aufmerksamkeitsleitendes Instrument bei der Analyse sehr großer Korpora. In: Bubenhofer, N. and Kupietz, M. (eds.): Visualisierung sprachlicher Daten: Visual Linguistics – Praxis – Tools. Heidelberg: Heidelberg University Publishing.

Rüdiger, J. O. (2018). CorpusExplorer v2.0 – Visualisierung prozessorientiert gestalten. In: Bubenhofer, N. and Kupietz, M. (eds.): Visualisierung sprachlicher Daten: Visual Linguistics – Praxis – Tools. Heidelberg: Heidelberg University Publishing.

Sinclair, J. (1991). Corpus, Concordance, Collocation. Oxford: Oxford University Press.

Teich, E., Fankhauser P. (2010). Exploring a Scientific Corpus Using Datamining. In: Gries, S. T., Wulff, S. and Davies, M. (eds.): Corpus-linguistic applications. Current studies, new directions. Amsterdam/New York, NY, 2010, VI, Rodopi, pp. 233-246

Teich, E., Degaetano-Ortlieb, S., Fankhauser, P., Kermes, H., and Lapshinova-Koltunski, E. (2014). The Linguistic Construal of Disciplinarity: A Data Mining Approach Using Register Features. Journal of the Association for Information Science and Technology (JASIST).

Tukey, J. W. (1977). Exploratory Data Analysis. Pearson.

Zeldes, A., Ritz, J., Lüdeling, A., and Chiarcos, C. (2009). ANNIS: A search tool for multi-layer annotated corpora. In Mahlberg, M., González-Díaz, V., and Smith, C., editors, Proceedings of the Corpus Linguistics 2009 Conference, Liverpool, UK.

Example-Based Querying for Linguistic Specialist Corpora

Roman Schneider
Institute for the German Language (IDS)
R5 6-16, 68161 Mannheim, Germany
schneider@ids-mannheim.de

Abstract

The paper describes preliminary studies regarding the usage of Example-Based Querying for specialist corpora. We outline an infrastructure for its application within the linguistic domain. Example-Based Querying deals with retrieval situations where users would like to explore large collections of specialist texts semantically, but are unable to explicitly name the linguistic phenomenon they look for. As a way out, the proposed framework allows them to input prototypical everyday language examples or cases of doubt, which are automatically processed by CRF and linked to appropriate linguistic texts in the corpus.

Keywords: Grammar and Syntax, Infrastructures and Architectures, Information Retrieval, Machine Learning Methods, Specialist Corpora

1. Introduction and Related Work

Specialist corpora do not only serve as data foundation for linguistic studies. Regarding the content aspect, they also provide invaluable access to research results reported in the corpus texts, and thus could be used to promote the transfer of knowledge from specialist corpora to individual learners. This relates in particular to monitor corpora of scientific journals, (virtual) collections of reference books for a certain domain, and archives comprising online information systems from the web.

The content of such information systems is sometimes technically stored as plain text, but more often as a combination of semantically structured XML-hypertexts and text-specific metadata. A prominent example for the linguistic domain is the grammatical information system *grammis*, hosted at the Institute for German Language (IDS) in Mannheim (IDS-Mannheim, 2018) (Schneider and Schwinn, 2014). It brings together terminological, lexicographical, bibliographical, and corpus-based information about German grammar, and combines the description of grammatical structures from a syntactic, semantic, or functional perspective with multimedia content such as graphics, sound, and animation (see figure 1). Thus, features of spoken language, the construction of morphological or syntactical structures, and the effects achieved by the transformation of these structures can be immediately illustrated. Other modules provide authentic datasets and empirical analyses of corpus studies on specific language phenomena, plus a scientific description of selected linguistic terminology with reference to corpus-based examples.

Among other things, such large hypertext collections – usually with a multitude of contributing authors – have to deal with terminological variety: The use of different vocabularies (i.e., terms or concepts that are specific for a certain approach) within documents can cause considerable difficulties for systematic content retrieval (Bubenhofer and Schneider, 2010) (Sharma and Mittal, 2016). This seems especially true for fields where theories or even authors tend to name comparable concepts differently, and where terminology often reflects miscellaneous needs of heterogeneous user groups.

As a consequence, a notorious problem of specialist corpora like *grammis* is the identification of appropriate content that suits the concrete question of the current user – often a search for the needle in a haystack. Apart from traditional retrieval utilities – (semantically enriched) full text search, keyword lists, table of contents etc. – we believe that natural language could play an important role in the exploration process, inasmuch as it allows users to gain accurate access to appropriate pieces of information without the need of learning specialized query languages or without the time-consuming task of filling out complex search forms. Moreover, it could offer a way-out in situations where users, due to terminological uncertainties, are unable to name a certain problem or the phenomenon they look for.

Related work exists for the underlying idea: The Linguist's Search Engine (LSE) tried to offer an "intuitive, linguistically sophisticated but user-friendly way" (Resnik and Elkiss, 2005) by adopting a strategy called "Query By Example" [1] for web searches based on POS tagging. TIGER Corpus Navigator allows users to navigate a corpus, based on example sentences that represent abstract linguistic concepts (Hellmann et al., 2010). Most recently, (Augustinus et al., 2012) and (Augustinus et al., 2016) introduced example-based treebank querying as a way to search within annotated corpus resources. They allow users to enter natural language sentences or phrase segments as a basis to search for similar syntactic constructions.

We expand this methodologically highly attractive idea in several ways: First, we apply it not on annotated treebank corpora, but on a heterogenous structured specialist corpus. Since the included texts provide information about natural language phenomena (object of investigation) with the help of natural language (means of communication), they consequently should be explorable with the same means of natural language. Second, we see example-based querying as an ideal way to open up scientific corpus resources to a broader public. Our focus is not restricted to users who lack experience in specialized corpus query languages, but also

[1] This approach should not be confused with a method of the same name for describing a database query strategy, originally developed by IBM (Zloof, 1977).

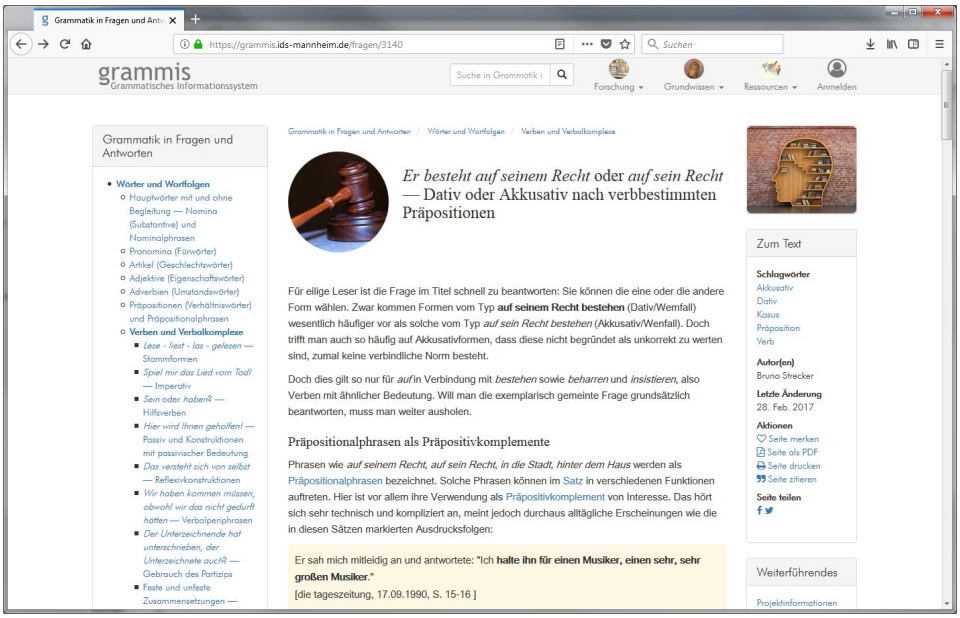


Figure 1: Online access to the grammis specialist corpus

on users with different terminological backgrounds, or even without explicit knowledge of linguistic terminology. The objective is to combine a language-oriented retrieval approach, which is supposed to be suitable for both linguists and linguistic laymen, with a data-oriented foundation.

This framework description is structured as follows: The next section introduces our proposed retrieval layers, as well as the resources they operate on. Section 3. covers the syntax-based layer – the place where example-based querying takes place – in more detail, featuring some prototypical examples. Section 4. summarizes the benefits and gives an outlook on ongoing work.

2. The Retrieval Environment

The *grammis* specialist corpus comprises nearly 3,000 XML-coded hypertext documents on grammatical topics that constitute a 4-million-token collection of specialized language. Furthermore, dictionaries of selected linguistic phenomena (like verb valency, genitive formation, connectors, affixes, prepositions) contribute about 1,500 textual entries with customized XML microstructures and an additional total of 2 million tokens. Both information types are valuable foundations for example-based querying, in the sense that they contain large quantities of natural language examples for illustration purposes, and that they are completely categorized by terminological keywords. Keywords are organized as controlled vocabularies – covering and interconnecting different linguistic theories and schools – within a terminology management system that features ISO-2788/ANSI Z39.19 compliant hyponymy/meronymy relationship types (Suchowolec et al., 2016).

The aggregated 4,500 XML units, containing a mixture of domain-specific specialist language and everyday language example sentences, constitute the overall search space. In order to facilitate content exploration, we consider the following resources:

- **Corpus of Tagged Examples:** Out of the specialist hypertexts and lexical entries, all XML-coded everyday language sentences are added to a corpus database of tagged examples. To enrich these approximately 5,500 samples with POS and morphological annotations about case, number, gender etc., we use the statistical tagger *MarMot* (Mueller et al., 2013), built upon Conditional Random Fields (CRF). For each example sentence, the corpus database stores a back reference to the source document and its corresponding keywords.

- **Dictionaries/Lexica:** The *grammis* lexical resources can be divided into ”flat” dictionaries, organized as simple word lists with attached explanatory texts, and ”enriched” dictionaries with explicitly coded semantic or syntactic content. The latter applies to the *E-VALBU* valency dictionary (IDS-Mannheim, 2010), which is based on the most comprehensive work on German verb valency – *VALBU* (Schumacher et al., 2004) — and includes detailed information about the arguments controlled by a verbal predicate. Furthermore, by adding the onomasiological classification from *Verben in Feldern (VIF)* (Schumacher, 1986), every verb can be assigned to a hierarchical set of semantically founded verb fields and subfields.

- **Terminological Net:** Another semantic resource utilized by the proposed retrieval framework comes out of the *grammis* terminology management system. The approximately 1,400 stored concepts form a poly-hierarchical network of meaningfully related specialized words, using standardised relationship types (syn-

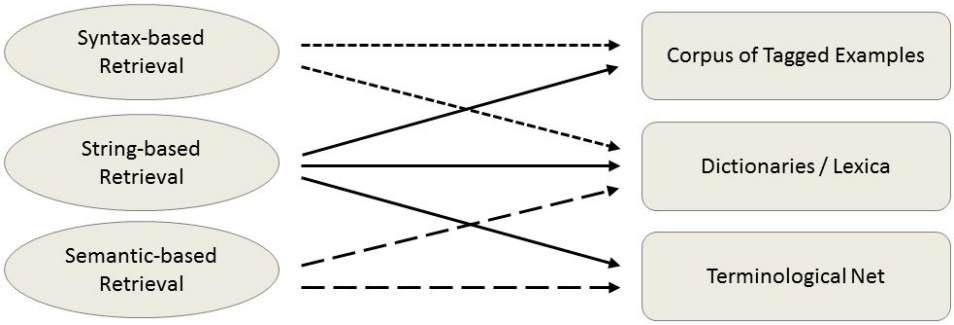


Figure 2: Retrieval Layers and Utilized Resources

onym, broader term, broader term partial etc.). They build the backbone for search strategies that handle user input containing specialist vocabulary. Within this resource, the plurality of linguistic theories is respected, language-specific and cross-linguistic categories are mapped (Haspelmath, 2016) (Moravcsik, 2016).

An ideal platform should allow the users to enter arbitrary free text, typically containing either terminological expressions and/or everyday language examples. This individual input will be automatically processed, using the resources listed above, and linked to corpus content addressing appropriate grammatical phenomena. Search is conducted using three independent layers (string-, semantic-, and syntax-based, see figure 2); their results are rated and merged after completion. [2]

2.1. String-Based Retrieval Layer

The string-based layer delivers reasonable search results in cases where the user enters string sequences that correspond to sequences within the information system's hypertext documents. Syntactically complete sentences as well as shorter phrases or even single words can be processed. The layer operates on the lexical and terminological texts, and looks for exact matches or string-based similarities. The algorithm uses character-based similarity measures, notably (i) Longest Common Substring (LCS), which computes the length of corresponding contiguous characters that exist in search and corpus strings (Gusfield, 1997), (ii) Damerau-Levenshtein for counting the minimum number of operations that are necessary to transform one string into another (Hall and Dowling, 1980), and (iii) Jaro-Winkler, which includes a prefix scale into the computation of number and order of common characters (Winkler, 1990).

[2]Rating and merging of the results from the three layers obviously represents another important procedural issue. We experimented with different weights, and had the impression that the top results of the string- or semantic-based layers in many cases outperformed the results of the syntax-based layer – provided that the former produced reasonable results at all. A best practice evaluation of measures is still pending.

2.2. Semantic-Based Retrieval Layer

This layer potentially adds corpus documents that are semantically related to input keywords, using the document's keyterms – either assigned manually or automatically via Automatic Term Extraction (ATE) (Suchowolec et al., 2017) – and the terminological net (see figure 3). Since terminological concepts are interlinked by multi-hierarchical relations, it is possible to determine semantic similarity by exploiting path length. Important knowledge-based values are the shortest path between two concepts and the maximum depth of the hierarchy (Leacock and Chodorow, 1998). For expanding search within the semantically annotated dictionaries, the layer also takes into account the lexical relations and the hierarchical set of semantically founded verb fields.

2.3. Syntax-Based Retrieval Layer

The syntax layer – which constitutes the heart of the example-based retrieval algorithm – takes over in cases where the user does not formulate his search inquiry terminologically, and where simple word-based lookups yield no satisfactory result. Instead, each user input is regarded as prototypical example sentence or phrase, and undergoes syntax-based processing.

In order to obtain an empirically determined test set, we collect typical everyday language queries, using an anonymized protocol of *grammis'* full text searches. We automatically filter out all requests that contain disambiguated grammatical key terms, and data containing less than three words. Out of the remaining $\sim$ 8,000 sentential expressions, we gradually build up a gold standard, performing manual filtering and double-blind indication of corresponding terminological keywords by human experts. In order to make this collection of typical user queries testable against models trained on the corpus of tagged examples, it is processed with the same morpho-syntactical tagging environment beforehand.

We arrange all computed metadata in a line-oriented CoNLL-like format (Hajič et al., 2009) that can be processed by CRF++ (Kudo, 2005 2013). Some real-life examples from the full text search test set are:

```
Ich PPER nom sg 0 1 0 0 0
habe VAFIN 0 sg 1 pres ind 0 0
mir PRF dat sg 1 0 0 0 0
```

```
oder KON 0 0 0 0 0 0 0
mich PPER acc sg 0 1 0 0 0
in APPR 0 0 0 0 0 0 0
die ART acc sg fem 0 0 0 0
Hand NN acc sg fem 0 0 0 0
geschnitten VVPP 0 0 0 0 0 0 0

sie PPER nom sg fem 3 0 0 0 0
leitet VVFIN 0 sg 3 pres ind 0 0
ein ART acc sg neut 0 0 0 0
pleites ADJA acc sg neut pos 0 0 0
Unternehmen NN acc sg neut 0 0 0 0
```

So after morpho-syntactic annotations are added, the layer operates on the enriched input dataset and tries to identify similar constructions. If this attempt is successful, it can either link directly to the corresponding corpus text, or identify semantically related texts by exploiting the keywords attached to the reference text.
A straightforward approach would initially look for exact syntactical equivalents, and then – if this generates too few results – ignore word order. We believe that the first variant works too restrictively in some situations, and that the second variant is too general and would often produce worthless results. So, if simple decision rules do not help, we argue that recourse to machine learning in general and – since we deal with sequential, word-oriented data potentially containing a broad set of morpho-syntactical metadata – CRF (Lafferty et al., 2001) in particular seems promising. With recourse to statistical methods, it can handle partial matches and situations where a syntax pattern is associated with different targets, and identify appropriate terminological keywords. Other possible metrics for comparing syntactic parse trees would be Tree Edit Distance, Tree Kernels, or Subtree Overlap.

3. Example-Based Querying at Work

We now focus on the example-based retrieval component, evaluating the syntax layer against naturally occurring searches.

3.1. Training

For the computation of a trained model file, we arrange all tagged sentences from the example corpus in the already mentioned line-oriented format. In every line, the first column contains a single token[3], the second column contains the corresponding POS tag, and the following columns represent morphological annotations. The last column shows the ID of an appropriate corpus text[4]:

```
Das ART nom sg neut 0 0 0 0 f3185
Land NN nom sg neut 0 0 0 0 f3185
Niedersachsen NE nom sg neut 0 0 0 0 f3185
wird VAFIN 0 sg 0 ind 3 0 pres f3185
sich PRF acc sg 0 0 3 0 0 f3185
nicht PTKNEG 0 0 0 0 0 0 0 f3185
an APPR 0 0 0 0 0 0 0 f3185
dem ART dat sg masc 0 0 0 0 f3185
europaweit ADJD 0 0 0 0 0 pos 0 f3185
autofreien ADJA dat sg masc 0 0 pos 0 f3185
Tag NN dat sg masc 0 0 0 0 f3185
am APPRART dat sg neut 0 0 0 0 f3185
Freitag, NN dat sg neut 0 0 0 0 f3185
den ART acc sg masc 0 0 0 0 f3185
22. ADJA acc sg masc 0 0 pos 0 f3185
September NN acc sg masc 0 0 0 0 f3185
beteiligen VVFIN 0 sg 0 ind 3 0 past f3185

Sie PPER nom sg fem 0 3 0 0 d312
hoffte, VVFIN 0 sg 0 ind 3 0 past d312
dass KOUS 0 0 0 0 0 0 0 d312
sie PPER nom pl * 0 3 0 0 d312
das PDS acc sg neut 0 0 0 0 d312
bis APPR 0 0 0 0 0 0 0 d312
zum APPRART dat sg masc 0 0 0 0 d312
Abend NN dat sg masc 0 0 0 0 d312
erledigt VVPP 0 0 0 0 0 0 0 d312
haben VAFIN 0 sg 0 ind 3 0 pres d312
wÃ¼rde VAFIN 0 sg 0 subj 3 0 pres d312
```

A template file describes which features should be used for the training run. Each line in the template specifies the involvement of certain metadata by addressing its relative position from the current token, e.g.:

```
# Unigram
U11:%x[-2,1]
U12:%x[-1,1]
U13:%x[0,1]
U14:%x[1,1]
U15:%x[2,1]

U20:%x[-2,2]
U21:%x[-1,2]
U22:%x[0,2]
U23:%x[1,2]
U24:%x[2,2]

U29:%x[-2,3]
U30:%x[-1,3]
U31:%x[0,3]
U32:%x[1,3]
U33:%x[2,3]

U38:%x[-2,4]
U39:%x[-1,4]
U40:%x[0,4]
U41:%x[1,4]
U42:%x[2,4]
```

Since each sentence is interlinked with one or more grammatical keywords and with one hypertext back reference, we distinguish between three training variants: (i) the last column contains a concatenation of all terms (ii) the last column contains only one selected term, as for example the highest/lowest ranking concept within the terminological

[3] Since string-based search is covered within a separate layer, we do not use tokens for the model training, and only consider the subsequent morpho-syntactical features.

[4] As described above, these texts have already been classified by linguistic keywords before.

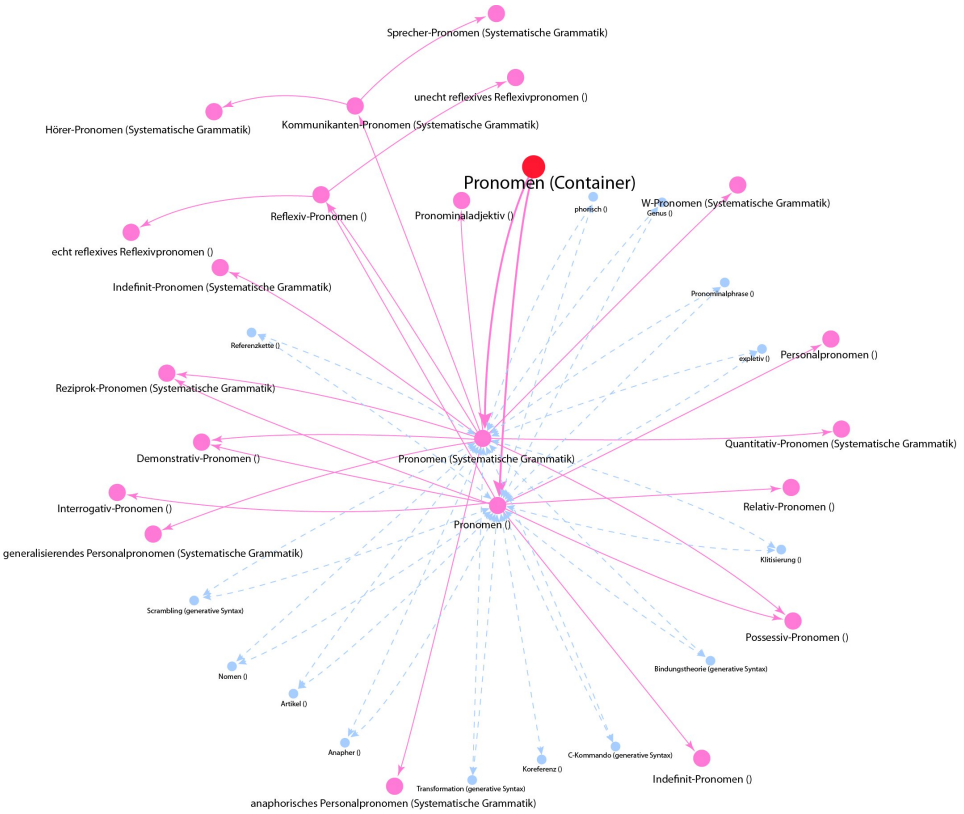


Figure 3: Partial visualization of the terminological net

net (iii) as in the training example above, the last column contains the back reference to a hypertext document, which in turn is annotated with one or more keywords.

3.2. Classification Testing

Out of our test set, we subsequently present two queries and use the trained model for retrieving appropriate explanatory hypertext documents. Work on the best adjustment of the classification model is still in its early stages; we will present a comprehensive evaluation after completion. Nethertheless, we believe that the following examples illustrate the fundamental suitability of example-based querying for the linguistic domain.

3.2.1. Use of Case for Date Specifications

Table 1 shows the tagged input of the first everday language input example (”am Freitag, den 13.”; English equivalent: ”on Friday, the 13th”). Obviously, the underlying – but not explicitely expressed – question concerns the correct use of case within a German date specification: Is the combination of dative and accusative acceptable, or should dative be maintained for the whole phrase (that would then be: ”am Freitag, dem 13.”)?

And indeed, when applying the back reference model as described above, the algorithm references a suitable explanatory corpus text containing similar example sentences. [5] The corresponding keywords of this document are *Akkusativ (accusative)*, *Dativ (dative)*, *Datum (date)*, *Deklination (declension)*, *Flexion (inflection)*, *Kasus (case)*.

Token	POS	Case	Num	Gen
am	APPRART	dat	sg	masc
Freitag	NN	dat	sg	masc
den	ART	acc	sg	masc
13.	ADV	acc	0	0

Table 1: First query example as CRF input

3.2.2. Use of Genitive Constructions

As a second example (”das Auto von meinem Vater”; English: ”the car of my father”), we choose an authentic user query that a human native speaker would probably classify as somehow related to the use of genitive constructions, although it does not contain any genitives at all (see table 2). A possible genitive construction would be ”meines Vaters Auto”; English: ”my father’s car”.

A syntactically similar example is found within a *grammis* hypertext on the use of the preposition ”von” and dative case, compared to the ”high-order” style of genitive attributes. Consequently, our classification algorithm generates an expedient link to this document. [6] Its classifying keywords are *Attribut (attribute)* and *Genitiv (genitive)*.

[5] https://grammis.ids-mannheim.de/fragen/3185

[6] https://grammis.ids-mannheim.de/fragen/4550

Token	POS	Case	Num	Gen
das	ART	nom	sg	neut
Auto	NN	nom	sg	neut
von	APPR	0	0	0
meinem	PPOSAT	dat	sg	masc
Vater	NN	dat	sg	masc

Table 2: Second query example as CRF input

4. Concluding Remarks

We proposed the intuitive and efficient use of example-based querying for content retrieval on a large collection of specialist hypertexts dedicated to linguistics. Overall, the results of the preliminary studies reveal the attractiveness of this preprocessing step for the thematic exploration of corpora containing natural language example sentences. When combined with string-based and semantic-based retrieval components, the proposed framework can assist users seeking qualified information in situations where they, due to terminological uncertainties, are unable to name the concrete problem they look for. In other words: the approach described in this article helps to transform object-related introductory questions, that are close to everyday language experience, into category-related retrieval questions.

We believe that example-based querying can also play an important role for the search in specialist corpora with different orientations, as long as they contain annotated natural language material whose morpho-syntactical structure is showing some noticeable characteristics. Possible examples range from the CosMov Corpora for Social Movement Research (*www.semtracks.org/cosmov/*) to the Corpus of American Soap Operas (*corpus.byu.edu/soap/*), but comprise even various types of historical corpora. The crucial point here is the corpus extension by metadata: In order to ensure that the syntax-based retrieval layer is able to identify semantically related texts, each corpus text has to be enriched – preferably automatically – with meaningful keyterms. It would be interesting to find out how additional metadata like the results of dependency parsing (Kübler et al., 2009) would enhance retrieval quality. Besides, example-based querying has already been evaluated for human motion data (Kim et al., 2016) and music retrieval using audio and fuzzy-music-sense features (Su et al., 2014).

Depending on keyword complexity and the number of annotation features, the described task also poses theoretical challenges to machine learning researchers, since different classification approaches seem appropriate. If the example sentences are associated with only one (rather general) keyword, queries generate quantitatively more, but mostly far too imprecise results – high recall, but low precision. Associating multiple keywords to every example sentence tends to produce higher error rates. Our tests indicate that using back references to terminologically classified corpus texts can be a satisfying trade-off.

In order to improve the retrieval quality of future *grammis* releases, we are planning to implement the described solution in conjunction with a fundamental extension of the system's content modules. Example-based querying will then be an important (pre-)processing step for the easy-to-use exploration of the large specialist corpus underlying the online information system. The terminological and ML-related resources will be made publicly available in order to foster follow-up research on example-based querying for natural language resources.

5. Bibliographical References

Augustinus, L., Vandeghinste, V., and Van Eynde, F. (2012). Example-based treebank querying. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC 2012)*. European Language Resources Association (ELRA).

Augustinus, L., Vandeghinste, V., and Vanallemeersch, T. (2016). Poly-GrETEL: Cross-lingual example-based querying of syntactic constructions. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Bubenhofer, N. and Schneider, R. (2010). Using a domain ontology for the semantic-statistical classification of specialist hypertexts. In *Papers from the Annual International Conference on Computational Linguistics (Dialogue)*, pages 622–628.

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, pages 1–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hall, P. A. V. and Dowling, G. R. (1980). Approximate string matching. *ACM Computing Surveys*, 12(4):381–402.

Haspelmath, M. (2016). The challenge of making language description and comparison mutually beneficial. *Linguistic Typology*, 20(2):299–304.

Hellmann, S., Unbehauen, J., Chiarcos, C., and Ngonga Ngomo, A.-C. (2010). The TIGER corpus navigator. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT-9)*, pages 91–102. Northern European Association for Language Technology (NEALT).

IDS-Mannheim. (2010). *Das elektronische Valenzwörterbuch deutscher Verben*. http://www.ids-mannheim.de/e-valbu/, DOI: 10.14618/evalbu.

IDS-Mannheim. (2018). *Grammis - Grammatisches Informationssystem*. https://grammis.ids-mannheim.de, DOI: 10.14618/grammis.

Kim, D., Jang, M., and Kim, J. (2016). Example-based retrieval system for human motion data. In *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pages 1–2, Sept.

Kübler, S., McDonald, R. T., and Nivre, J. (2009). *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Kudo, T. (2005 – 2013). CRF++: Yet Another CRF Tool Kit. Version 0.58.

Lafferty, J. D., McCallum, A., and Pereira, F. C. (2001). Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 282–289.

Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. In *WordNet: An electronic lexical database*, pages 265–283. MIT Press, Cambridge, MA, USA.

Moravcsik, E. (2016). On linguistic categories. *Linguistic Typology*, 20(2):417–426.

Mueller, T., Schmid, H., and Schütze, H. (2013). Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA. Association for Computational Linguistics.

Resnik, P. and Elkiss, A. (2005). The linguist's search engine: An overview. In *Proceedings of the ACL 2005 Interactive Poster and Demonstration Session*, pages 33–36. Association for Computational Linguistics (ACL). DOI: 10.3115/1225753.1225762.

Schneider, R. and Schwinn, H. (2014). Hypertext, Wissensnetz und Datenbank: Die Web-Informationssysteme grammis und Progr@mm. In Franz Josef Berens et al., editors, *Ansichten und Einsichten. 50 Jahre Institut für Deutsche Sprache*, pages 337–346. IDS, Mannheim. http://nbn-resolving.de/urn:nbn:de:bsz:mh39-24719.

Schumacher, H., Kubczak, J., Schmidt, R., and de Ruiter, V. (2004). *VALBU - Valenzwörterbuch deutscher Verben*. Number 31 in Studien zur Deutschen Sprache. Narr, Tübingen.

Schumacher, H. (1986). *Verben in Feldern. Valenzwörterbuch zur Syntax und Semantik deutscher Verben*. Number 1 in Schriften des Instituts für Deutsche Sprache. de Gruyter, Berlin / New York.

Sharma, V. and Mittal, N. (2016). Cross lingual information retrieval (CLIR): Review of tools, challenges and translation approaches. *Information System Design and Intelligent Application*, pages 699–708.

Su, J. H., Wang, C. Y., Chiu, T. W., Ying, J. J. C., and Tseng, V. S. (2014). Semantic content-based music retrieval using audio and fuzzy-music-sense features. In *2014 IEEE International Conference on Granular Computing (GrC)*, pages 259–264, Oct.

Suchowolec, K., Lang, C., and Schneider, R. (2016). Re-designing online terminology resources for german grammar. In Philipp Mayr, et al., editors, *NKOS 2016 Networked Knowledge Organization Systems Workshop. Proceedings of the 15th European Networked Knowledge Organization Systems Workshop (NKOS 2016)*, pages 59–63. http://nbn-resolving.de/urn:nbn:de:bsz:mh39-52982.

Suchowolec, K., Lang, C., Schneider, R., and Schwinn, H. (2017). Shifting complexity from text to data model. adding machine-oriented features to a human-oriented terminology resource. In J. Gracia, et al., editors, *Language, Data, and Knowledge. Springer Lecture Notes in Artificial Intelligence*, pages 203–212. DOI: 10.1007/978-3-319-59888-8.

Winkler, W. E. (1990). String comparator metrics and enhanced decision rules in the Fellegi-Sunter model of record linkage. In Philipp Mayr, et al., editors, *Proceedings of the Section on Survey Research Methods. American Statistical Association*, pages 354–359.

Zloof, M. M. (1977). Query-by-example: A data base language. *IBM Systems Journal*, 16(4):324–343.

Increasing Interoperability for Embedding Corpus Annotation Pipelines in Wmatrix and other corpus retrieval tools

Paul Rayson
School of Computing and Communications, Lancaster University, Lancaster, UK
p.rayson@lancaster.ac.uk

Abstract

Computational tools and methods employed in corpus linguistics are split into three main types: compilation, annotation and retrieval. These mirror and support the usual corpus linguistics methodology of corpus collection, manual and/or automatic tagging, followed by query and analysis. Typically, corpus software to support retrieval implements some or all of the five major methods in corpus linguistics only at the word level: frequency list, concordance, keyword, collocation and n-gram, and such software may or may not provide support for text which has already been tagged, for example at the part-of-speech (POS) level. Wmatrix is currently one of the few retrieval tools which have annotation tools built in. However, annotation in Wmatrix is currently limited to the UCREL English POS and semantic tagging pipeline. In this paper, we describe an approach to extend support for embedding other tagging pipelines and tools in Wmatrix via the use of APIs, and describe how such an approach is also applicable to other retrieval tools, potentially enabling support for tagged data.

Keywords: corpus annotation, tagging, API

1. Introduction

Many different computational tools are used to support corpus linguistics research. Typically, these fit into one of three main categories. First, *compilation* tools are used to support corpus collection, and these include transcription, OCR, scanning and encoding tools. In the web-as-corpus paradigm, this category also includes web scraping and cleaning tools to collect data from web pages, online forums or social media along with tools to remove boilerplate or duplicated text, e.g. WebBootCaT (Baroni et al., 2006). Second, once a corpus is compiled, it may need to be *annotated* at one or more levels for later linguistic analysis. A common such level is part-of-speech (POS) annotation which has proved fruitful over the years for grammatical analysis and as a stepping stone to other higher levels such as semantic tagging. Annotation may be applied manually by one person or collaboratively by a team (e.g. using such tools as eMargin[1] or Brat[2]), and/or automatically using pre-existing tagging software (e.g. CLAWS (Garside and Smith, 1997)). The final category of corpus software is the most often used and cited in corpus papers (see Rayson (2015) for a quantitative in-depth survey), that of corpus *retrieval*. Corpus retrieval software began life[3] around 50 years ago with computerised concordances in key-word-in-context (KWIC) format, and steadily gained extra features such as frequency lists, keywords, collocations and n-grams. Recent developments, notably demonstrated by many papers at the Corpus Linguistics 2017 conference in Birmingham, have been to bring in tools and methods from other areas such as Natural Language Processing, such as topic modelling, or for researchers to develop their own software, or use other scripting languages (such as Python or R) to carry out analyses (as pioneered by Baayen (2008) and Gries (2013)). In this paper, we restrict ourselves to the pre-existing and widely available corpus query engines and retrieval tools.

[1] https://emargin.bcu.ac.uk/

[2] http://brat.nlplab.org/

[3] See http://timemapper.okfnlabs.org/muranava/history-of-computerised-corpus-tools for an excellent visual time line produced by Mura Nava.

2. Limitations of existing retrieval tools

One important limitation with many corpus retrieval tools is their ability to deal only with raw unannotated corpora and provide results only at the word level. This reduces the power of queries to surface patterns in the text and fails to take advantages of lemma searches which depends on POS analysis to link surface forms to dictionary headwords. Rayson (2008) and Culpeper (2009) have also shown the advantages of performing keyness analysis beyond the level of the word by combining the key words approach pioneered by Scott (1997) with semantic annotation. Workarounds with regular expressions do permit some of the existing desktop corpus query tools (such as WordSmith and AntConc) to work with tagged data, but it is the web-based corpus retrieval systems such as BYU (Davies, 2005) and CQPweb (Hardie, 2012) which have sufficient storage, power and complexity to more fully exploit tagged corpora.

The second major restriction, even with some existing web-based retrieval systems, is that corpus data must be tagged before it can be loaded (if the tool supports upload of new data directly) in or indexed by these tools. Only a few tools combine corpus annotation tools with corpus retrieval methods, and for ease of use by non-technical users, this combination offers many advantages and a shallow learning curve. Sketch Engine[4] incorporates POS taggers and lemmatisers for many languages and text is automatically tagged after upload or during processing of web-derived corpora. LancsBox (Brezina et al., 2015) version 3 also now incorporates the TreeTagger[5] in order to POS tag and lemmatise some languages. Wmatrix, through its Tag Wiz-

[4] https://www.sketchengine.co.uk/

[5] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

ard feature permits the upload and automatic POS and semantic tagging of English corpora.

3. Improving Interoperability

Corpus linguists have noted that different tools produce different results e.g. even in terms of calculating the size of a corpus (Brezina and Timperley, 2017) due to tokenisation differences resulting from a programmer's decisions encoded in software or by corpus compilers methods. For reproducibility and replication purposes there are many advantages to be gained from comparing and retaining separate implementations of standard corpus methods. However, corpus software development has now reached a level of maturity where good software development practices of design, implementation, distribution and component reuse should be adopted.

Some previous research into interoperability of corpus methods has been limited and small scale, and focussed on potential quick wins for linking analysis components in a small group of web-based tools (Wmatrix, CQPweb, IntelliText, and WordTree) (Moreton et al., 2012). By connecting such tools together, we are not just improving interoperability and reusability, but this will enable researchers to try out research methods and tools that are established in other disciplinary communities but are not so familiar in their own. For example, there are many similar tools to those developed in corpus linguistics which have long been employed in other areas that are not so well known to corpus researchers, from at least three other areas: (a) Computer Assisted Qualitative Data Analysis (CAQDAS) tools such as ATLAS.ti, Nvivo, Wordstat (b) Psycholinguistics software such as Linguistic Inquiry and Word Count (LIWC) and (c) Digital Humanities tools such as Voyant and MONK.

Amongst other work on interoperability is an ongoing effort to develop a Corpus Query Lingua Franca (CQLF) ISO standard[6] for corpus query formatting although this is not adopted in any query tools, and Vidler and Wattam (2017) have proposed a metadata standard for describing properties of corpus files and resources to enable better sharing and reuse of data between tools. In the remainder of the paper, we focus on addressing limitations related to the lack of flexible annotation facilities in corpus query tools. Other options for interoperability and reproducibility would be to share retrieval method modules directly.

4. Linking corpus annotation pipelines to retrieval tools

In order to improve interoperability between tagging pipelines and retrieval tools, we propose the use of Application programming interfaces (APIs)[7]. An API can be created to send raw corpus data to a remote server, where it is tagged, and then return the result. Such an approach will enable support for taggers and tagging pipelines to be incorporated not only into web-based corpus retrieval tools, but also in downloadable desktop applications. Web-based software is hosted on more powerful servers where taggers could more easily be housed alongside retrieval systems but for downloadable software such as AntConc and WordSmith, a user's personal computer may not be powerful enough to run large tagging processes in reasonable amounts of time. Even with web-based systems on remote servers, there may be a requirement to run extremely large tagging jobs in other parallel systems such as high performance clusters or Hadoop/SPARK Map Reduce frameworks (Wattam et al., 2014). This is where an API-based approach has important advantages since it enables the separation of the corpus processing and is independent of platform so existing Linux-based taggers can be linked for users running Windows locally, for example.

APIs have already been embedded in a small number of corpus tools to link with the compilation phase rather than the annotation phase. For example, in Laurence Anthony's AntCorGen tool[8], he has implemented an API link to the PLOS One research database to enable searching and download of academic papers which can be turned into a corpus. Also, Laurence's FireAnt[9] employs the Twitter Streaming API similarly to download Tweets and collate them into a corpus for later analysis.

4.1. Wmatrix case study

Wmatrix[10] is a corpus software tool combining annotation and retrieval methods. It provides a web interface to the existing USAS semantic tagger and CLAWS part-of-speech tagger corpus annotation tools, and standard corpus linguistic methodologies such as frequency lists, key words and concordances. It also extends the keywords method to key grammatical categories and key semantic domains by combining the taggers with the keyness metric (Rayson, 2008). Wmatrix allows the non-technical user to run these tools in a tag wizard via a web browser such as Chrome, and so will run on any computer (Mac, Windows, Linux, Unix) with a network connection. Earlier versions were available for Unix via terminal-based command line access (tmatrix) and Unix via X-windows (Xmatrix), but these only offered retrieval of text pre-annotated with USAS and CLAWS. With the incorporation of the two taggers, Wmatrix was designed for the analysis of English data only and has been used for a wide range of projects from learner data, interview transcript analysis, fiction and non-fiction corpora.

Wmatrix includes multiple components written in different programming languages, see Figure 1 for the architecture diagram of the current system. The two taggers, CLAWS and USAS, are written in C. The frequency profiling, concordancing and keyness comparison tools are also written in C. The collocation tool is developed in Java. Unix shell scripts and Perl scripts act as glue to link all these components together to the web front end. Underlying the system, the corpus data is stored in a Unix file system. User access is currently controlled using Apache's basic authentication (htaccess). The current version, Wmatrix3, is hosted on Lancaster University's cloud infrastructure on a Debian virtual machine.

[6] https://www.iso.org/standard/37337.html?browse=tc

[7] https://en.wikipedia.org/wiki/Application_programming_interface

[8] http://www.laurenceanthony.net/software/antcorgen/

[9] http://www.laurenceanthony.net/software/fireant/

[10] http://ucrel.lancs.ac.uk/wmatrix/

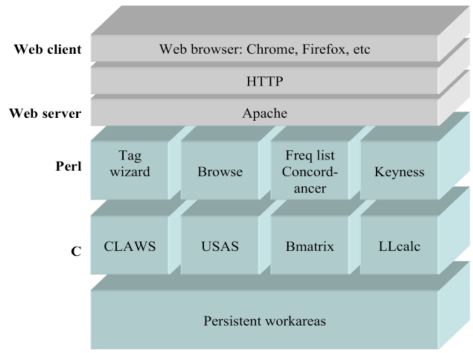


Figure 1: Wmatrix architecture.

A key and imminent requirement is to allow Wmatrix to be used with languages other than English. Semantic taggers already exist for a number of other languages (Russian and Finnish) and in recent years, we have been bootstrapping semantic taggers for an increasing number of languages (Piao et al., 2015; Piao et al., 2016; Piao et al., 2017a) and historical time periods (Piao et al., 2017b). Linguistic resources for these semantic taggers are freely available on GitHub[11] and many of them have been made available via a SOAP API and a downloadable GUI[12]. Our proposal is to create REST APIs for these and other tools[13] and add a new module into the existing Wmatrix architecture which sends data to be tagged via these APIs. This would sit alongside the existing tag wizard and remove the need to install other taggers directly on the Wmatrix server alongside the CLAWS and USAS modules. Non-English data which cannot be tagged through the existing tag wizard will then be directed to these components sitting on other servers for tagging. If we retain the two levels of annotation (POS and semantic tagging) then the existing infrastructure of Wmatrix will be sufficient, however further levels of annotation (e.g. dependency parsing) will require additional database or indexing operations to be implemented.

Once this architecture is in place, sample code will be made available on the API server to allow other corpus retrieval tools to access the taggers. Similar APIs could be used to permit other pipelines such as Stanford Core, GATE and OpenNLP to be linked. Further development of desktop tools would be needed for them to become annotation-aware once they can tag data via the APIs. In terms of feasibility of implementation of the REST APIs and incorporation into Wmatrix, this is fairly low risk. We have been running and supporting the CLAWS web based tagger since 1998, and the SOAP APIs for the multilingual USAS taggers for three years, and APIs have become widely adopted for NLP analytics tools and websites.

[11] https://github.com/UCREL/Multilingual-USAS

[12] http://ucrel.lancs.ac.uk/usas/

[13] to be hosted on http://ucrel-api.lancs.ac.uk/

5. Conclusion

In this paper, we have outlined a proposal for enabling interoperability between two major types of corpus linguistics software, the annotation and retrieval (or query) systems. Currently, there are only a handful of tools which incorporate some form of both types, and previous research has shown the benefits of corpus queries operating beyond the level of the word. To improve ease of use for non-technical users, we propose the embedding and linking of further corpus annotation pipelines so that end-users can add annotation to their own data and then continue to use their current preferred tools for research. This proposed approach will contribute to improved interoperability in the corpus software community by simplifying the addition of new methods to existing tools and is complementary to other efforts to foster exchangeable and reusable components across corpus platforms. Other potential options, such as installing the taggers locally (rather than Wmatrix itself), can be explored, but it is expected that the API route would be preferable to tool developers and end-users alike.

6. Acknowledgements

Wmatrix was initially developed within the REVERE project (REVerse Engineering of Requirements) funded by the EPSRC, project number GR/MO4846, 1998-2001. Ongoing maintenance of taggers (e.g. Linux porting work by Stephen Wattam), development of new components (e.g. L-gram developed by Eddie Bell and C-grams developed by Andrew Stone) and dictionary updates (e.g. by Sheryl Prentice) are funded by user licence fees. Semantic taggers for new languages are being developed by Scott Piao and funded by the UCREL research centre, in collaboration with a large number of researchers worldwide[14]. Metaphor extensions have been developed in the MELC project (Metaphor in end-of-life care) funded by the ESRC (grant reference ES/J007927/1). The Historical Thesaurus Semantic Tagger (HTST) was developed in the SAMUELS project (Semantic Annotation and Mark-Up for Enhancing Lexical Searches) funded by the AHRC in conjunction with the ESRC (grant reference AH/L010062/1). The Welsh Semantic Tagger research is funded by the UK Economic and Social Research Council (ESRC) and Arts and Humanities Research Council (AHRC) as part of the CorCenCC (Corpws Cenedlaethol Cymraeg Cyfoes; The National Corpus of Contemporary Welsh) Project (Grant Number ES/M011348/1).

7. Bibliographical References

Baayen, R. H. (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics using R*. Cambridge University Press, Cambridge.

Baroni, M., Kilgarriff, A., Pomikálek, J., and Rychlý, P. (2006). WebBootCaT: a web tool for instant corpora. In *Proceedings of Euralex*, Turin.

Brezina, V. and Timperley, M. (2017). How large is the BNC? a proposal for standardised tokenization and word counting. In *Proceedings of Corpus Linguistics 2017*, Birmingham, UK.

[14] See http://ucrel.lancs.ac.uk/usas/ for details.

Brezina, V., McEnery, T., and Wattam, S. (2015). Collocations in context: A new perspective on collocation networks. *International Journal of Corpus Linguistics*, 20(2):139–173.

Culpeper, J. (2009). Keyness: words, parts-of-speech and semantic categories in the character-talk of shakespeare's romeo and juliet. *International Journal of Corpus Linguistics*, 14(1):29–59.

Davies, M. (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.

Garside, R. and Smith, N. (1997). A hybrid grammatical tagger: CLAWS4. In Roger Garside, et al., editors, *Corpus Annotation: Linguistic Information from Computer Text Corpora*, pages 102–121. Longman, London.

Gries, S. T. (2013). *Statistics for linguistics with R*. Mouton De Gruyter, Berlin and New York, 2nd edition.

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

Moreton, E., Nesi, H., Rayson, P., Sharoff, S., and Stephenson, P. (2012). Corpus tools interoperability survey. Technical report.

Piao, S., Bianchi, F., Dayrell, C., D'egidio, A., and Rayson, P. (2015). Development of the multilingual semantic annotation system. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 1268–1274. Association for Computational Linguistics, 6.

Piao, S., Rayson, P., Archer, D., Bianchi, F., Dayrell, C., El-Haj, M., Jiménez, R.-M., Knight, D., Křen, M., Löfberg, L., Nawab, R., Shafi, J., Teh, P., and Mudraya, O. (2016). Lexical coverage evaluation of large-scale multilingual semantic lexicons for twelve languages. In Nicoletta Calzolari, et al., editors, *Proceedings of LREC 2016, Tenth International Conference on Language Resources and Evaluation*, pages 2614–2619. European Language Resources Association (ELRA).

Piao, S., Rayson, P., Knight, D., Watkins, G., and Donnelly, K. (2017a). Towards a Welsh semantic tagger: creating lexicons for a resource poor language. In *Proceedings of Corpus Linguistics 2017*.

Piao, S., Dallachy, F., Baron, A., Demmen, J., Wattam, S., Durkin, P., McCracken, J., Rayson, P., and Alexander, M. (2017b). A time-sensitive historical thesaurus-based semantic tagger for deep semantic annotation. *Computer Speech and Language*, 46:113–135.

Rayson, P. (2008). From key words to key semantic domains. *International Journal of Corpus Linguistics*, 13(4):519–549.

Rayson, P. (2015). Computational tools and methods for corpus compilation and analysis. In Douglas Biber et al., editors, *The Cambridge Handbook of English corpus linguistics*, pages 32–49. Cambridge University Press.

Scott, M. (1997). PC analysis of key words - and key key words. *System*, 25(2):233–245.

Vidler, J. and Wattam, S. (2017). Keeping properties with the data: CL-MetaHeaders - an open specification. In Piotr Bański, et al., editors, *Proceedings of the Workshop on Challenges in the Management of Large Corpora and Big Data and Natural Language Processing (CMLC-5+BigNLP) 2017 including the papers from the Web-as-Corpus (WAC-XI) guest section. Birmingham, 24 July 2017*, pages 35–41.

Wattam, S., Rayson, P., Alexander, M., and Anderson, J. (2014). Experiences with parallelisation of an existing nlp pipeline: tagging hansard. In Nicoletta Calzolari, et al., editors, *Proceedings of LREC 2014, Ninth International Conference on Language Resources and Evaluation*, pages 4093–4096. ELRA.