Challenges in the Management of Large Corpora (CMLC-2)

Workshop Programme

14:00 – 14:10 – Introduction

14:10 – 14:30
Marc Kupietz, Harald Lüngen, Piotr Bański and Cyril Belica,
Maximizing the Potential of Very Large Corpora: 50 Years of Big Language Data at IDS Mannheim

14:30 – 15:00
Adam Kilgarriff, Pavel Rychlý and Miloš Jakubíček,
Effective Corpus Virtualization

15:00 – 15:30
Dirk Goldhahn, Steffen Remus, Uwe Quasthoff and Chris Biemann,
Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web

15:30 – 16:00
Vincent Vandeghinste and Liesbeth Augustinus,
Making a large treebank searchable online. The SONAR case.

16:00 – 16:30 Coffee break

16:30 – 17:00
John Vidler, Andrew Scott, Paul Rayson, John Mariani and Laurence Anthony,
Dealing With Big Data Outside Of The Cloud: GPU Accelerated Sort

17:00 – 17:30
Jordi Porta,
From several hundred million words to near one thousand million words: Scaling up a corpus indexer and a search engine with MapReduce

17:30 – 17:50
Hanno Biber and Evelyn Breiteneder,
Text Corpora for Text Studies. About the foundations of the AAC-Austrian Academy Corpus

17:50 – 18:00 – Closing remarks

Editors

Marc Kupietz	Institut für Deutsche Sprache, Mannheim
Hanno Biber	Institute for Corpus Linguistics and Text Technology, Vienna
Harald Lüngen	Institut für Deutsche Sprache, Mannheim
Piotr Bański	Institut für Deutsche Sprache, Mannheim
Evelyn Breiteneder	Institute for Corpus Linguistics and Text Technology, Vienna
Karlheinz Mörth	Institute for Corpus Linguistics and Text Technology, Vienna
Andreas Witt	Institut für Deutsche Sprache, Mannheim and University of Heidelberg
Jani Takhsha	Institut für Deutsche Sprache, Mannheim

Workshop Organizers

Marc Kupietz	Institut für Deutsche Sprache, Mannheim
Hanno Biber	Institute for Corpus Linguistics and Text Technology, Vienna
Harald Lüngen	Institut für Deutsche Sprache, Mannheim
Piotr Bański	Institut für Deutsche Sprache, Mannheim
Evelyn Breiteneder	Institute for Corpus Linguistics and Text Technology, Vienna
Karlheinz Mörth	Institute for Corpus Linguistics and Text Technology, Vienna
Andreas Witt	Institut für Deutsche Sprache, Mannheim and University of Heidelberg

Workshop Programme Committee

Lars Borin	University of Gothenburg
Dan Cristea	"Alexandru Ioan Cuza" University of Iasi
Václav Cvrček	Charles University Prague
Mark Davies	Brigham Young University
Tomaž Erjavec	Jožef Stefan Institute, Ljubljana
Alexander Geyken	Berlin-Brandenburgische Akademie der Wissenschaften
Andrew Hardie	University of Lancaster
Nancy Ide	Vassar College
Miloš Jakubíček	Lexical Computing Ltd.
Adam Kilgarriff	Lexical Computing Ltd.
Krister Lindén	University of Helsinki
Jean-Luc Minel	Université Paris Ouest Nanterre La Défense
Christian Emil Ore	University of Oslo
Adam Przepiórkowski	Polish Academy of Sciences
Uwe Quasthoff	Leipzig University
Pavel Rychlý	Masaryk University Brno
Roland Schäfer	FU Berlin
Marko Tadić	University of Zagreb
Dan Tufiş	Romanian Academy, Bucharest
Tamás Váradi	Hungarian Academy of Sciences, Budapest

Table of contents

Author Index

Introduction

We live in an age where the well-known maxim that “the only thing better than data is more data” is something that no longer sets unattainable goals. Creating extremely large corpora is no longer a challenge, given the proven methods that lie behind e.g. applying the Web-as-Corpus approach or utilizing Google's n-gram collection. Indeed, the challenge is now shifted towards dealing with large amounts of primary data and much larger amounts of annotation data. On the one hand, this challenge concerns finding new (corpus-)linguistic methodologies that can make use of such extremely large corpora e.g. in order to investigate rare phenomena involving multiple lexical items, to find and represent fine-grained sub-regularities, or to investigate variations within and across language domains; on the other hand, some fundamental technical methods and strategies are being called into question. These include e.g. successful curation of data, management of collections that span multiple volumes or that are distributed across several centres, methods to clean the data from non-linguistic intrusions or duplicates, as well as automatic annotation methods or innovative corpus architectures that maximise the usefulness of data or allow to search and to analyze it efficiently. Among the new tasks are also collaborative manual annotation and methods to manage it as well as new challenges to the statistical analysis of such data and metadata.

Maximizing the Potential of Very Large Corpora: 50 Years of Big Language Data at IDS Mannheim

Marc Kupietz, Harald Lüngen, Piotr Bański, Cyril Belica
Institute for the German Language (IDS)
R5 6–13, 68161 Mannheim, Germany
corpuslinguistics@ids-mannheim.de

Abstract

Very large corpora have been built and used at the IDS since its foundation in 1964. They have been made available on the Internet since the beginning of the 90's to currently over 30,000 researchers world-wide. The Institute provides the largest archive of written German (Deutsches Referenzkorpus, DeReKo) which has recently been extended to 24 billion words. DeReKo has been managed and analysed by engines known as COSMAS and afterwards COSMAS II, which is currently being replaced by a new, scalable analysis platform called KorAP. KorAP makes it possible to manage and analyse texts that are accompanied by multiple, potentially conflicting, grammatical and structural annotation layers, and is able to handle resources that are distributed across different, and possibly geographically distant, storage systems. The majority of texts in DeReKo are not licensed for free redistribution, hence, the COSMAS and KorAP systems offer technical solutions to facilitate research on very large corpora that are not available (and not suitable) for download. For the new KorAP system, it is also planned to provide sandboxed environments to support non-remote-API access "near the data" through which users can run their own analysis programs.[†]

Keywords: very large corpora, scalability, big data

1. History of corpora and corpus technology at the IDS

While the IDS was founded in 1964, at least from 1967, under the directors Paul Grebe and Ulrich Engel, a department called *Documentation of the German language* was in place, in which a text collection of contemporary German was compiled and recorded on punchcards (Teubert and Belica, 2014, p.300). The first electronic corpus to be released was the Mannheimer Korpus I (MK I, 1969), which comprised 2.2 million words in 293 texts of mostly fiction, including some popular fiction, and newspaper text. In 1972, a smaller, additional part called MK II with more text types was added. In the subsequent project *Grundstrukturen der deutschen Sprache*, the MK data were extensively analysed and used as the empirical basis in 17 published volumes on grammatical themes between 1971 and 1981 (Teubert and Belica, 2014, 301). Over the years, more corpora were added, amongst other things from branches of the IDS which were hosting specific projects, such as the Bonner Zeitungskorpus (Hellmann, 1984). At that time, the corpus data were maintained by the computing centre of the IDS, and linguists had to specify their queries to programmers who would then formulate them in machine-readable form. Between 1982 and 1992, the first proprietary concordancer REFER was in use at the IDS computing centre. REFER supported interactive, sentence-oriented queries in up to 17 million running words including basic grammatical categories, e.g. verb and adjective inflection. In 1991, the project COSMAS (Corpus Search, Management and Analysis System) was launched with the goal of developing an integrated corpus platform and research environment that would enable linguists at the IDS to formulate and refine their queries to the IDS text collections flexibly and independently at their own personal computer. From the beginning, the COSMAS developers subscribed to a set of innovative corpus linguistic methodological principles which are still widely acknowledged in current corpus technology, amongst other things the concept of a unified data model, of multiple concurring linguistic annotations, and most of all the introduction of the concept of a *virtual corpus*. A virtual corpus is a sample of texts drawn from the complete text collection according to a problem-specific view described in terms of external (e.g. bibliographic, specified in terms of metadata) or internal (e.g. the distribution of certain keywords, search patterns or annotations) criteria. Consequently, as of 1992, when the software was first deployed, the COSMAS system offered the tools by means of which users could define their own virtual corpora such that they were representative or balanced w.r.t to their own specific research questions, as well as save or possibly publish them (cf. al Wadi, 1994, p. 132ff).

The successor project COSMAS II was launched in 1995, and from 1996, COSMAS could be used via the internet by linguists all over the world. A part of the project had also been concerned with acquiring more text data, and in 1998, the project *DeReKo I – Deutsches Referenzkorpus* (German reference corpus) started as a cooperation with the universities of Stuttgart and Tübingen. One of its achievements was a mass acquisition of newspaper, fictional, and other text types from publishing houses and individuals, and by the end of the project in 2002, DeReKo contained 1.8 billion tokens. Since then, *Deutsches Referenzkorpus* has been retained as the name of all written corpus holdings at the IDS. By 2004, the IDS corpus extension project had been made a permanent project, and in 2012, DeReKo reached the size of 5 billion word tokens. Since 2008, the IDS has also been a partner in the national and European research infrastructure initiatives TextGrid, D-SPIN, and CLARIN, in which the concept of virtual corpora has been extended and imple-

[†]The authors would like to thank Michael Hanl and and Nils Diewald for their help in preparing the present contribution.

mented to encompass location-independent *virtual collections* (van Uytvanck, 2010).

2. Recent developments

- we organize our acquisition campaigns in waves, addressing 50 to 200 potential license/text donors at a time
- in addition, we approach publishers (in particular the relevant licensing departments) directly at book fairs and sometimes on the phone
- in the negotiations, we seek to acquire licenses as liberal as possible for scientific use, in order of priority: CLARIN-ACA, QAO-NC, QAO-NC-LOC (see Kupietz and Lüngen, 2014)
- chances of convincing rights holders to donate licenses are on average 5%
- the expenses for the acquisition and curation of one word of fictional text are presently around 25,000 times higher than the expenses for one word of newspaper text (see Kupietz, 2014)
- considering only the regularly incoming text data according to existing license agreements, the current growth rate of DeReKo is 1.7 billion words per year

Table 1: Corpus acquisition trivia

As a result of recent campaigns and acquisition deals, DeReKo has grown to over six billion word tokens until 2013, and further grown by a factor of four in the first half of 2014, now containing more than 24 billion word tokens. In the following, we shortly describe the major recent contributions – for more details see Kupietz and Lüngen (2014).

Wikipedia is an example of a web corpus that can be curated under a sufficiently liberal license, and we have made available all German Wikipedia articles and talk pages in DeReKo twice in 2011 (Bubenhofer et al., 2011) and 2013 (Margaretha and Lüngen, in preparation). The 2013 conversion, for example, amounts to more than 1 billion word tokens.
In cooperation with the PolMine project of the University of Duisburg-Essen[1], we have adapted all debate protocols of parliaments in Germany (national and state level) since around the year 2000 (comprising around 360 million word tokens), and we continue to curate the protocols from previous years and other German-speaking parliaments.
We have also curated around 6 million words of fictional text as a result of our 2011 campaign addressing publishers of fiction, while more is in the conversion pipeline.

The bulk of the increase in DeReKo in 2014, however, is formed by a large news database archive for which we obtained DeReKo-specific rights from its commercial provider, containing 102 million documents of press text, specialized journals and specialized e-books. For the latest DeReko release in 2014, we have prepared and included the press data part, containing 98 national and regional newspapers and magazines starting between 2000 and 2003, which amounted to more than 16 billion new word tokens. As a result, the latest DeReKo release contains more than 24 billion word tokens and takes up 550 GB of disk space without annotations (see 2). The new data not only increase the size but also the disperson of genres and topics in DeReKo (see Kupietz and Lüngen, 2014).

- the DeReKo corpus archive (without annotations and version history) uses 550 GB disk space in 1500 files
- for corpus processing, we currently use a machine with 48 cores and 256 GB RAM running CentOS 5.10
- all corpus processing is done in a massively parallel fashion
- for pre-processing of raw data, we use Perl scripts, pdf2html, TagSoup, and tidy
- for the main processing, we use the Saxon Enterprise Edition XSLT-2.0/3.0-Processor
- for coarse quality control, we use Adam Kilgarriff's (2001) measure for corpus similarity
- the POS annotation of the entire DeReKo archive requires between 1 and 2 CPU months for each tool
- the annotation of near-duplicate clusters (mostly carried out within same-source corpora only) (Kupietz, 2005) takes about 7 CPU days
- deriving dependency annotation of the entire DeReKo archive requires between 2 CPU months with Xerox XIP and 13 CPU years with MATE (estimated value, based on a 2.5% DeReKo sample)
- the inter-annotation-tool-agreement on POS tags is typically around 91.5% (see Belica et al., 2011, for details)
- the primary data of DeReKo have been version-controlled and stored in a Subversion repository (currently using 130 GB storage) since 2007
- all DeReKo releases, including primary and annotation data, have been redundantly archived on off-line storage devices since 2009
- long-term preservation and metadata export for OAI-PMH (OAI-PMH, 2008) is currently being migrated to our centre for the long-term preservation of German linguistic research data (Fankhauser et al., 2013)

Table 2: Corpus processing trivia

3. Big Data?

"Big Data" is a broad and fuzzy term, allowing for numerous particular interpretations. Whether it is taken to mean an amorphous mixture or simply an extremely large amount of data of some specific kind, the Deutsches Referenzkorpus DeReKo fulfils both definitions: the latter in a straightforward way, given its current size and growth (see Figure 1), and the former thanks to its status as a primordial sample, from which users can draw virtual corpora (see section 1. and Kupietz et al., 2010).
Figure 2 shows that, measured by the required number of units of the contemporary portable storage medium, the amount of memory needed for the primary text data of DeReKo was actually highest in the beginning in 1969 (factor 400), then decreased to a level of around factor 1 in 1992, where it has remained since then. Only the storage require-

[1] http://polmine.sowi.uni-due.de/

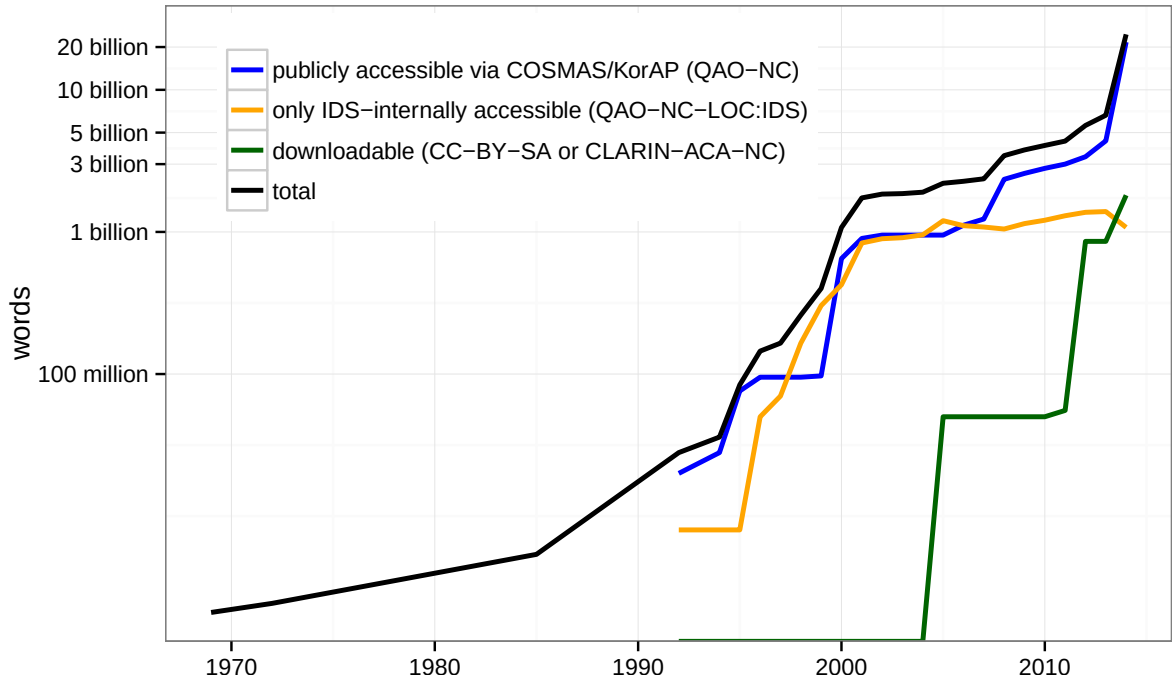


Figure 1: Development of the size of DeReKo in words and its accessibility since 1969 (see Kupietz and Lüngen, 2014, for explanations of the license-abbreviations).

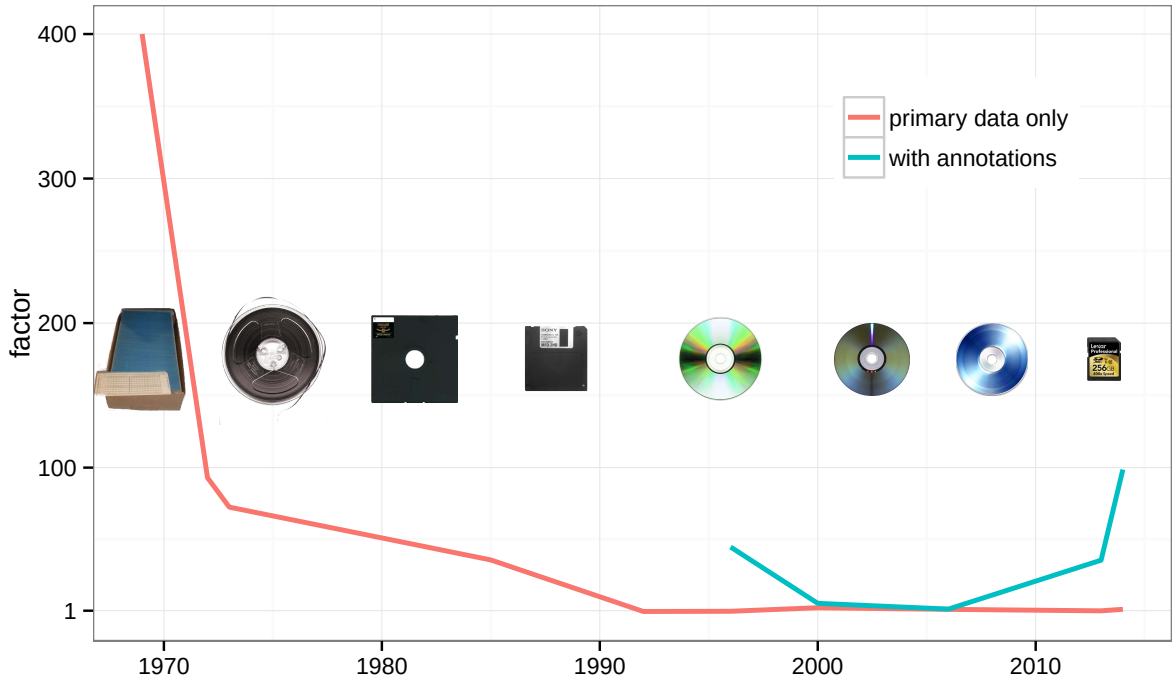


Figure 2: Development of DeReKo's storage requirements since 1969 in relation to contemporary portable storage media. (The figures until 1993 are estimated. For 1969 a box with 2000 punchcards was taken as reference.

ments of the linguistic annotations provided for DeReKo have increased as of 2008, i.e. after the introduction of three new taggers for linguistic annotation (Belica et al., 2011), to a factor of almost 100 of current storage units in 2014 Hence, if we consider the corpus sizes over the years in relation to the available portable storage media, it turns out that the IDS has actually dealt with "big data" ever since the introduction of language corpora.

4. Licensing very large corpora or: "Why don't you just make it open-access?"

Almost as long as its tradition of constructing corpora and making them available to the scientific community is the IDS's tradition of being the notorious bearer of – and sometimes the convenient scapegoat for – the bad news: DeReKo cannot be made available for download. The simple reason

is not that the IDS is a "bad center" that likes sitting on "its data". The reason is of course that the IDS – just like every other provider of comparable corpora – does not have the right to to make the data available for download and that the acquisitions of such a right, e.g. via national licenses, for corpora of the size of DeReKo or a fraction thereof, would hardly be within the limitations of public funding (cf. Kupietz et al., 2010, p. 1849). To our relief, however, this fact is now becoming more and more common ground in the community.[2] This is largely due to the generally increased awareness of intellectual property rights and other personal rights thanks to the Google Books discussion and possibly also thanks to the educational work carried out within CLARIN, often invoking more or less desperate analogies to confront the lack of understanding, such as, e.g., Paweł Kamocki's (2013) analogy with Victor Lustig's selling of the Eiffel Tower in 1925, the image of the car-industry being 'anti-science' due to not providing linguists with free cars, the tightrope that corpus providers walk on (Kupietz, 2009), the cowboy who prefers to shoot first (Ketzan and Kamocki, 2012), or the one with Lady Justice and the balance of interests (next paragraph).

In any case, thanks to this development, approaches aiming at improving the situation for researchers without interfering with the equally legitimate interests of rights holders (on whose donations the researchers vitally depend after all) can nowadays be discussed more openly. As sketched in Figure 3 (Kupietz, 2010), there are more factors involved in such a typical balance of interests and most of them can be gradual. Accordingly, there are many toeholds for such improvements (including the attachment of the scale, in analogy to the legal framework) and to achieve the best results, ideally all of them should be taken into account. One of the very promising approaches is to extend the stack of "technical precautions" in such a way that the possible types of use can be extended to include most types of research that would otherwise only be possible if the corpus were available for download. Part of our current work in this direction is sketched in section 5.2..

Apart from such technical measures along the lines of Jim Gray's (2003) "put the computation near the data" (see also Kupietz et al., 2010; Bański et al., 2012), in our acquisition campaigns, we always try and have always tried to negotiate licenses that are as open as the respective rights holder allows it without the stack of "money" growing too high (see Table 1). Open licenses are great, but what to do if a rights holder does not want to sign them, even if you have been quite inventive to put everything you have on his scale pan?

5. Accessing DeReKo: COSMAS and KorAP

At present, DeReKo is accessible via the Corpus Search, Management and Analysis System COSMAS II (al Wadi, 1994; Bodmer Mory, 2014). It is currently used by more than 32,000 registered users and can handle a DeReKo part of about 7-8 billion words (in one archive) with up to 2 morphosyntactic annotation layers. Due to its having been designed already in the early nineties, its scalability has now reached its limits, because it depends, for example, on holding all indices in RAM and because there are currently no solutions to distribute the system over multiple machines. Because of the above limitations, in 2011 we started the project KorAP (Bański et al., 2012, 2013, 2014), to develop a new corpus analysis platform from scratch, aiming at a scalable and extensible scientific tool, sustainable for a lifecycle of 20 years. Since January 2014, KorAP has been open for IDS-internal alpha testing.

5.1. Scalability

One of the major aims for KorAP has been to achieve horizontal scalability, in order to support a theoretically unlimited number of tokens with a theoretically unlimited number of annotation layers built upon those tokens. This is why KorAP features a multi-component architecture communicating via a simple REST web interface, thus allowing all services to run on different physical machines. The system supports a web UI to allow users to conveniently browse all available data, as well as a policy management component to provide data entry points for restricted resources. These components furthermore include a module to serialize query information (Bański et al., 2014) as well as two search backends that are responsible for processing the query and for retrieving the search results. In order to ensure sustainability, all the components are interchangeable. The backends are based on well-proven Open Source search technologies (Lucene/Solr and Neo4j), which support distributed searches, in this way ensuring horizontal scalability.

5.2. Bringing the computation near the secured data

Another fundamental aim of KorAP was to maximize the research potential on copyright-protected texts. Given the growing amount of textual data and annotations, as well as the growing complexity of the relations among the data components, traditional security measures appear to be inadequate for the task of protecting the integrity of the data while at the same time allowing for fine-grained access control of selected texts and/or selected annotation layers.

KorAP implements a flexible management system to encompass access scenarios that have been identified in the planning phase of the project. That system uses a separate data store to hold security policies for resources that may be subject to restrictions, either imposed by licence agreements concerning the texts and/or the products of annotation tools, or imposed by the users themselves, when they choose to share self-defined virtual collections, potentially containing self-created annotations.

The two backends store DeReKo data in index structures for fast retrieval. Between the web client (the frontend or the API) and KorAP only search requests and results are transmitted. Policy management handles access to different levels of information (raw corpus data, annotation layers, virtual collections, etc.) and authentication based on a ses-

[2] Ironically, however, a slight step backwards was triggered by the open-access movement involving some confusion concerning the actual rights-holders of corpus texts in contrast to the rights-holders of research data in other disciplines, where the rights belong to the researchers themselves or at least to the scientific community.

Figure 3: In the provision and use of corpora, the collision of the basic rights: freedom of science and research, and guarantee of property, in practice, boils down to a balance of interests between researchers and rights holders.

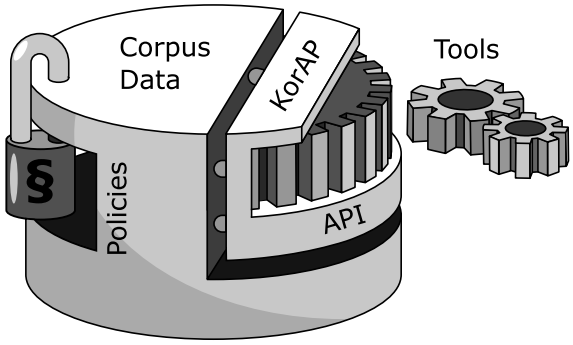


Figure 4: KorAP exposes corpus data for external tools under the control of access policies

sion model (as an intermediate layer between the API entry point and the corpus data).
In addition to standard API access, work is planned on making it possible to create sandboxed environments to support non-remote-API access "near the data", to allow direct access on license protected data and, in general, to provide access to big data without causing unnecessary, if not impossible, network traffic.

6. Conclusions

Written language corpora have been compiled in "big data" dimensions at the IDS since its inception in 1964, and due to the need of curating, storing, annotating, administrating, publishing, and querying them, their continuous expansion has necessarily always been accompanied by pioneering research in the areas of corpus linguistics methodology and corpus technology. The German Reference Corpus DeReKo, as the IDS written corpora collection has been called since the beginning of the millennium, has reached the size of 24 billion word tokens or 30 TB (including linguistic annotations) in 2014, due to the constant influx of text data according to previous license agreements as well as recent acquisitions. Given these dimensions, the scalability capacity of the present corpus management system COSMAS II, which was designed already at the beginning of the 90s, has reached its limits, and a new system called KorAP has been created to ensure continual access to the IDS data.
KorAP aims at horizontal scalability to support an arbitrary number of tokens provided with an arbitrary number of annotation layers. To maximize the research potential on copyright protected texts, it will provide a non-remote API for user-supplied mobile applications.

7. References

al Wadi, D. (1994). *COSMAS - Ein Computersystem für den Zugriff auf Textkorpora*. Institut für Deutsche Sprache.

Bański, P., Diewald, N., Hanl, M., Kupietz, M., and Witt, A. (2014). Access Control by Query Rewriting: the

Case of KorAP. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. European Language Resources Association (ELRA).

Bański, P., Fischer, P. M., Frick, E., Ketzan, E., Kupietz, M., Schnober, C., Schonefeld, O., and Witt, A. (2012). The New IDS Corpus Analysis Platform: Challenges and Prospects. In Calzolari, N., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul. European Language Resources Association (ELRA).

Bański, P., Frick, E., Hanl, M., Kupietz, M., Schnober, C., and Witt, A. (2013). Robust corpus architecture: a new look at virtual collections and data access. In Hardie, A. and Love, R., editors, *Corpus Linguistics 2013 Abstract Book*, pages 23–25, Lancaster. UCREL. http://ucrel.lancs.ac.uk/cl2013/doc/CL2013-ABSTRACT-BOOK.pdf.

Belica, C., Kupietz, M., Lüngen, H., and Witt, A. (2011). The morphosyntactic annotation of DeReKo: Interpretation, opportunities and pitfalls. In Konopka, M., Kubczak, J., Mair, C., Šticha, F., and Wassner, U., editors, *Selected contributions from the conference Grammar and Corpora 2009*, pages 451–471, Tübingen. Gunter Narr Verlag.

Bodmer Mory, F. (2014). Mit COSMAS II »in den Weiten der IDS-Korpora unterwegs«. In Steinle, M. and Berens, F. J., editors, *Ansichten und Einsichten. 50 Jahre Institut für Deutsche Sprache*, page 376–385. Institut für Deutsche Sprache, Mannheim.

Bubenhofer, N., Haupt, S., and Schwinn, H. (2011). A comparable Wikipedia corpus: From Wiki syntax to POS Tagged XML. In Hedeland, H., Schmidt, T., and Wörner, K., editors, *Multilingual Resources and Multilingual Applications. Proceedings of the Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, volume 96B of *Working Papers in Multilingualism*, pages 141–144, Hamburg. Hamburg University.

Fankhauser, P., Fiedler, N., and Witt, A. (2013). Forschungsdatenmanagement in den Geisteswissenschaften am Beispiel der germanistischen Linguistik. *Zeitschrift für Bibliothekswesen und Bibliographie (ZfBB)*, 60(6):296–306.

Gray, J. (2003). Distributed Computing Economics. Technical Report MSR-TR-2003-24, Microsoft Research.

Hellmann, M. W., editor (1984). *Ost-West-Wortschatzvergleiche. Maschinell gestützte Untersuchungen zum Vokabular von Zeitungstexten aus der BRD und der DDR*, volume 48 of *Forschungsberichte des Instituts für deutsche Sprache*. Narr, Tübingen.

Kamocki, P. (2013). Legal Issues: A checklist for data practitioners. Talk given at the EUDAT Workshop on DARUP on 2013-09-23 in Barcelona.

Ketzan, E. and Kamocki, P. (2012). CLARIN-D: Legal and Ethical Issues. Talk given at the Universität des Saarlandes, 2012-03-28.

Kilgarriff, A. (2001). Comparing corpora. *International Journal of Corpus Linguistics*, 6(1):97–133. http://www.kilgarriff.co.uk/Publications/2001-K-CompCorpIJCL.pdf.

Kupietz, M. (2005). Near-Duplicate Detection in the IDS Corpora of Written German. Technical Report kt-2006-01, Institut für Deutsche Sprache. ftp://ftp.ids-mannheim.de/kt/ids-kt-2006-01.pdf.

Kupietz, M. (2009). 45 years of walking the tightrope. Talk given at the D-SPIN/CLARIN-workshop on legal issues 2009-09-21 in Berlin.

Kupietz, M. (2010). Legal and Ethical Issues with respect to LRT and e-infrastructures. Talk given at the D-SPIN Scientific Board Meeting on 2010-12-10 in Berlin.

Kupietz, M. (2014). Der Programmbereich Korpuslinguistik am IDS: Gegenwart und Zunkunft. In Steinle, M. and Berens, F. J., editors, *Ansichten und Einsichten. 50 Jahre Institut für Deutsche Sprache*, page 298–319. Institut für Deutsche Sprache, Mannheim.

Kupietz, M., Belica, C., Keibel, H., and Witt, A. (2010). The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, page 1848–1854, Valletta, Malta. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2010/pdf/414_Paper.pdf (25.5.2010).

Kupietz, M. and Lüngen, H. (2014). Recent Developments in DeReKo. In *Proceedings of LREC 2014*. European Language Resources Association (ELRA).

Margaretha, E. and Lüngen, H. (in preparation). Building linguistic corpora from wikipedia articles and discussions.

Teubert, W. and Belica, C. (2014). Von der Linguistischen Datenverarbeitung am IDS zur Mannheimer Schule der Korpuslinguistik. In Steinle, M. and Berens, F. J., editors, *Ansichten und Einsichten. 50 Jahre Institut für Deutsche Sprache*, page 320–328. Institut für Deutsche Sprache, Mannheim.

van Uytvanck, D. (2010). CLARIN Short Guide on Virtual Collections. Technical report, CLARIN. http://www.clarin.eu/files/virtual_collections-CLARIN-ShortGuide.pdf.

Effective Corpus Virtualization

Miloš Jakubíček[‡†], **Adam Kilgarriff**[†], **Pavel Rychlý**[‡†]
‡ NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic
† Lexical Computing Ltd., Brighton, United Kingdom
jak@fi.muni.cz,adam@lexmasterclass.com,pary@fi.muni.cz

Abstract

In this paper we describe an implementation of corpus virtualization within the Manatee corpus management system. Under *corpus virtualization* we understand logical manipulation with corpora or their parts grouping them into new (virtual) corpora. We discuss the motivation for such a setup in detail and show space and time efficiency of this approach evaluated on a 11 billion word corpus of Spanish.

Keywords: corpus, corpus linguistics, virtualization, indexing, database

1. Introduction

This paper brings together two notions from widely separated areas: *virtualization* and *corpora*. Text corpora – large collections of electronic texts – are one of the essential resources for linguistics and have a firm place within computational linguistics and natural language processing. They have applications in a wide range of fields, providing reliable evidence for linguists and statistical models for engineers.

Virtualization has become one of the technology buzzwords of the beginning millenium as more and more people were seeking for appropriate solution for managing large scale IT resources. They were in place and ready to be used but often the inability to carry out reliable predictions that would help distributing the resources among the related services turned out to be a big problem. Using a resource is always a commitment, within the IT field much related to money investments and so questions like: "Does this server need 2, 4, 8 gigs of memory and the other one less, or vice versa, or should we just buy more?" gained a lot of importance, since the right answer led to large savings.

Since better predictions were not really available on the dynamic IT market, people aimed at a technological solution that would allow them to postpone their commitments or not to do them at all; and so we soon witnessed the take up of virtualization starting with processor and memory virtualization allowing a single physical system to host a number of virtual ones and distribute resources among them, continuning with storage virtualization and finally creating a sole market of cloud services.

Current situation in corpus linguistics is to some extent similar to that in IT before virtualization: for many languages there are large text collections available (see e.g. (Jakubíček et al., 2013a; Callan et al., 2009; Pomikálek et al., 2012)) and one has to decide how these will be organized into corpora at the technical level, i.e. as independent database units resulting from a (possibly costly, both in terms of runtime and final space occupation) indexing procedure.

While we presume that the corpus axiom is: the bigger the better, clearly having just a single huge corpus per language is not always desirable for obvious practical reasons – smaller data is always faster to process, one corpus implies one annotation scheme which would then be very limited, and finally one might just find himself in a situation where the subject of studies would be a portion of the language (possibly defined using complex constraints).

The obvious solution to this problem lying in creating separate and independent corpora for any combination of needs becomes less and less feasible for very large datasets. Therefore in this paper we would like to introduce the concept of corpus virtualization, a method allowing flexible management of corpora into logical units, as implemented within the Manatee corpus management system (Rychlý, 2007) used in the Sketch Engine (Kilgarriff et al., 2004).

The structure of the paper is as follows: in the next section we briefly describe the Manatee corpus management system and its past approaches to corpus organization, then we present the approach based on virtualization and its evaluation on a sample dataset.

2. Manatee

Manatee (Rychlý, 2000; Rychlý, 2007) is an all-in-one corpus management system specifically designed for text corpora. As any database system its elementary components can be divided into those that are used for compiling (indexing, building) the corpus (database) index files and those that are then used to query the corpus. Manatee uses a sophisticated set of index files based on the well-known inverted-index approach (Knuth, 1997) allowing complex but fast searching even for complex annotations using the Corpus Query Language (CQL, see(Jakubíček et al., 2010)).

Any reasonable indexing of text data starts with providing an efficient string-to-number mapping of the input words (or lemmas, or tags, etc.) as described in (Jakubíček et al., 2013b). The resulting data structure is called a *lexicon* and allows all other indices to operate on numbers, not on strings, and therefore to be smaller and faster to use.

The corpus consists of three elementary entities: *attributes* (such as word, lemma, tag), *structures* (sentences, paragraphs, documents) and *structure attributes* (metadata on structures, such as document ID), where for any attribute the following indices are compiled:

- attribute text (IDs in the order of appearance in the corpus)
- inverted index (list of positions for each ID)
- lexicon (string $\leftrightarrow$ ID mapping)

corpus	number of tokens (billions)	database size (gigabytes)
esAmTenTen11	8.7	217
esEuTenTen11	2.4	35
esTenTen11	11.1	252

Table 1: Overview of the esTenTen corpus and its parts.

Corpus parts are managed by specifying *subcorpora* for a corpus, where a subcorpus is simply defined (both conceptually and at the technical level) as a set of corpus segments (based e.g. on meta-data annotation). A subcorpus cannot be used as a standalone corpus, it is always accessed only from the main corpus. The subcorpus compilation creates just one index file specifying the subcorpus segments and as for query evaluation, the subcorpus serves just as a filter on top of the full corpus indices.

3. Corpus Virtualization

A *virtual corpus* is defined as a set of segments from one or more corpora. A virtual corpus therefore might be used just for a subcorpus as well if the segments originate from a single corpus – but in most cases this will not be the case and a virtual corpus will rather be a *supercorpus* in this respect. It uses the very same configuration setup as any regular corpus in Manatee except that instead of specifying input text to be processed, a definition file for virtual corpus is given in the following simple format:

```
=bnc
0,1000000
10000000,11000000
=susanne
0,$
=brown
0,1000
```

Each line starting with an equal sign specifies a source corpus to be used, otherwise lines are comma separated position pairs denoting segments to be included into the virtual corpus (where a dollar sign means last corpus position).
This definition file would describe a virtual corpus consisting of the first and eleventh million tokens of the BNC corpus and the whole Susanne corpus and the first 1,000 tokens from the Brown corpus.
While the subcorpus can be seen as a very light-weight concept, a virtual corpus is a heavy-weight mechanism and virtual corpora are first-class databases – they can be accessed without any knowledge of where they come from.
Compilation of a virtual corpus consists mainly of providing a new lexicon and mappings to all existing lexicons of the source corpora. Apart of that only the preexisting indices of those corpora are used for query evaluation resulting in large storage savings while having negligible influence on the query evaluation performance.
We demonstrate the advantages of virtual corpora as opposed to the regular ones on the example of the Spanish esTenTen corpus (Kilgarriff and Renau, 2013) consisting of two substantial parts, the esEuTenTen (European Spanish) and esAmTenTen (American Spanish) as given in Table 1. In Table 2 a comparison of its compilation in both variants is provided: as a virtual corpus consisting of two regular corpora and as a single regular corpus. As can be seen, the virtual corpus approach achieves space savings by factor of more than 20 and time saving by factor of more than 10.

	virtual	regular
space occupied	13 GB	252 GB
compilation time	3.4 hrs	30.6 hrs

Table 2: Comparison of the esTenTen being compiled as a virtual and regular corpus.

4. Related Work

Similar approach to the management issues of large text corpora has been taken within the COSMAS project focusing on the German Reference Corpus (DEREKO, (Kupietz et al., 2010)) where the concept of a virtual corpus suits for selecting working texts out of the whole DEREKO corpus, which would correspond to the subcorpus concept within Manatee. To the authors' best knowledge, the presented approach is unique in that the virtualization operates on entirely independent database entities and the virtualization process creates such a database as result too.

5. Conclusions and Further work

In this paper we justified and presented a method for corpus virtualization within the Manatee corpus management system that is efficient in terms of both savings in compilation time and occupied disk space while having very little footprint on the system performance during query evaluation.
Another exploitation of corpus virtualization currently under development is that for effective paralellization of corpus compilation by dividing the data into n-parts that will be compiled separately, then joined into a virtual corpus and this corpus will be finally devirtualized into a regular one.
All the implemented functionality belongs to the open source part of the Manatee corpus management system released as Manatee-open under the GPLv2 software license at `http://nlp.fi.muni.cz/trac/noske`.

6. References

Callan, J., Hoy, M., Yoo, C., and Zhao, L. (2009). Clueweb09 data set. Online presentation available at: `http://boston.lti.cs.cmu.edu/classes/11-742/S10-TREC/TREC-Nov19-09.pdf`.

Jakubíček, M., Rychlý, P., Kilgarriff, A., and McCarthy, D. (2010). Fast Syntactic Searching in Very Large Corpora for Many Languages. In *PACLIC 24 Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation*, pages 741–747, Tokyo.

Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., and Suchomel, V. (2013a). The TenTen Corpus Family. In *7th International Corpus Linguistics Conference CL 2013*, pages 125–127, Lancaster.

Jakubíček, M., Šmerk, P., and Rychlý, P. (2013b). Fast construction of a word-number index for large data. In A. Horák, P. R., editor, *RASLAN 2013 Recent Advances in Slavonic Natural Language Processing*, pages 63–67, Brno. Tribun EU.

Kilgarriff, A. and Renau, I. (2013). esTenTen, a Vast Web Corpus of Peninsular and American Spanish . *Procedia - Social and Behavioral Sciences* , 95(0):12 – 19. Corpus Resources for Descriptive and Applied Studies. Current Challenges and Future Directions: Selected Papers from the 5th International Conference on Corpus Linguistics (CILC2013) .

Kilgarriff, A., Rychlý, P., Smrž, P., and Tugwell, D. (2004). The Sketch Engine. In *Proceedings of the Eleventh EURALEX International Congress*, pages 105–116, Lorient, France. Universite de Bretagne-Sud.

Knuth, D. E. (1997). Retrieval on secondary keys. *The art of computer programming: Sorting and Searching*, 3:550–567.

Kupietz, M., Belica, C., Keibel, H., and Witt, A. (2010). The German Reference Corpus DeReKo: A Primordial Sample for Linguistic Research. In Nicoletta Calzolari (Conference Chair) and Khalid Choukri and Bente Maegaard and Joseph Mariani and Jan Odijk and Stelios Piperidis and Mike Rosner and Daniel Tapias, editor, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, pages 1848–1854, Valletta, Malta, May. European Language Resources Association (ELRA).

Pomikálek, J., Jakubíček, M., and Rychlý, P. (2012). Building a 70 billion word corpus of English from ClueWeb. In Nicoletta Calzolari (Conference Chair) and Khalid Choukri and Thierry Declerck and Mehmet Uğur Doğan and Bente Maegaard and Joseph Mariani and Jan Odijk and Stelios Piperidis, editor, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May. European Language Resources Association (ELRA).

Rychlý, P. (2000). *Korpusové manažery a jejich efektivní implementace*. PhD Thesis, Masaryk University, Brno.

Rychlý, P. (2007). Manatee/Bonito - A Modular Corpus Manager. In *1st Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 65–70, Brno.

Top-Level Domain Crawling for Producing Comprehensive Monolingual Corpora from the Web

Dirk Goldhahn[1], Steffen Remus[2], Uwe Quasthoff[1], Chris Biemann[2]
[1]University of Leipzig, Leipzig, Germany
[2]Technische Universität Darmstadt, Darmstadt, Germany
{dgoldhahn, quasthoff}@informatik.uni-leipzig.de, {remus, biem}@cs.tu-darmstadt.de

Abstract

This paper describes crawling and corpus processing in a distributed framework. We present new tools that build upon existing tools like Heritrix and Hadoop. Further, we propose a general workflow for harvesting, cleaning and processing web data from entire top-level domains in order to produce high-quality monolingual corpora using the least amount of language-specific data. We demonstrate the utility of the infrastructure by producing corpora for two under-resourced languages. Web corpus production for targeted languages and/or domains thus becomes feasible for anyone.

Keywords: corpus creation, web crawling, map reduce, web-scale corpora

1. Introduction

With the extraordinary growth of information in the World Wide Web, online documents increasingly become the major source for creating high quality corpora. Unfortunately, the development of technologies that make the information conveniently available to the user causes the process of crawling language data to become even harder. That is why researchers more and more rely on data provided by companies specialized in crawling the web, with all limitations that go along with this (cf. [7]).

We present an approach for creating corpora from the web with only little effort and by using only freely available, open-source software. All components used for data processing can be executed in a highly distributed environment, resulting in quick processing times. Researchers, data analysts and others are hence able to create large-scale high quality corpora targeted towards their own needs. In a case study, we will create two corpora for under-resourced languages, Kiswahili and Faroese. We discuss potential pitfalls, and ways to avoid them.

While there has been a number of initiatives in the past to obtain very large monolingual web corpora, for example WaCky[1] [1], COW[2] [11], Leipzig Corpora Collection [10], or even the very comprehensive *common crawl*[3] provided by Amazon, our contribution lies a) in the comprehensiveness for low-resource languages reached with minimal effort by crawling entire top-level domains, b) in the generic distributed processing pipeline for arbitrary automatic annotations and c) in the availability of the entire processing chain as open-source software component – partially provided by us.

The article is structured as follows: Section 2 describes the proposed approach to crawling and pre-filtering of entire top-level domains. Section 3 presents a generic distributed processing pipeline, which allows us to process very large amounts of data, and Section 4 gives detailed information regarding the availability of the presented tools and the gathered data during the case study described in Section 5. Section 6 summarizes and concludes this work.

2. Crawling

For crawling, we rely on the *Heritrix* project[4] (Version 3.1.1). The Heritrix archival crawler project is an open-source, web-scale crawler made available by the Internet Archive Community. It is used e.g. for periodically creating snapshots of large amounts of webpages in the web, which corresponds to the scheme of creating corpora from the web. *Heritrix* is a versatile tool, providing many options to configure the desired crawling behavior. Compared with other crawling software like *wget*, *HTTrack*, or *Nutch*, it offers several general advantages: Single crawl jobs can cover hundreds of millions of pages; it is stable, fast and follows more links than other comparable tools due to better handling of Java-script links while it is still easy to use.

Heritrix is initialized with a list of specified webpages – called seed – from which it extracts web links to other webpages that are subsequently downloaded and processed accordingly. Here, we will use it to harvest entire Top Level Domains (TLD), which means we download every suited web document we encounter in a particular TLD. The initially provided list of up to 2,000 seed domains for each TLD contains randomly chosen URLs coming from previous crawls. The composition of the seed has only minor influence on the results of the crawling process: Typically, hubs of a TLD – i.e. websites that contain links to a many different websites – are reached within the first steps of the process. We configured Heritrix to extract links from URLs and to follow them while not leaving the current TLD and not downloading the same URL twice or more.

[1] http://wacky.sslmit.unibo.it/
[2] http://hpsg.fu-berlin.de/cow/
[3] http://commoncrawl.org/
[4] http://crawler.archive.org

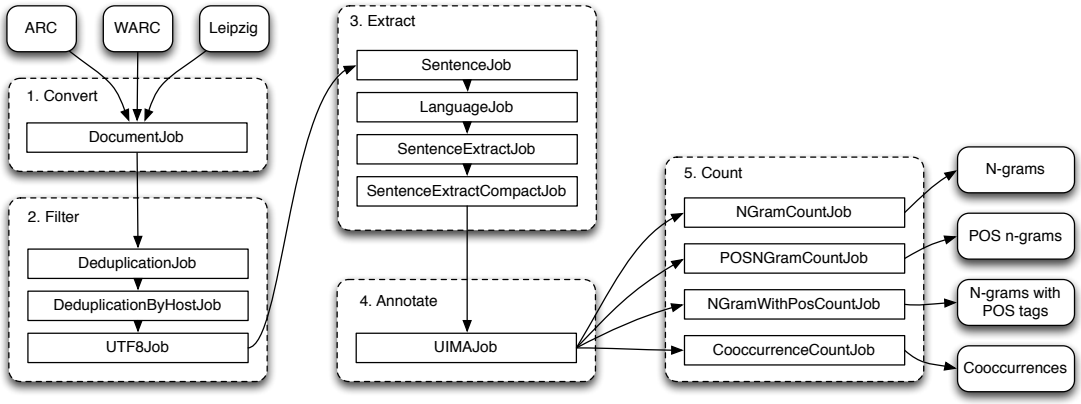


Figure 1: The individual jobs in the standard WebCorpus pipeline. Figure taken from [3].

Running on a machine with 8 CPU-cores and 20GB of RAM using an unthrottled 1GBit/s Internet connection, it reaches crawling speeds of up to 200 URLs/s for each crawling job while running up to 3 jobs in parallel. We chose to reduce load for single servers by limiting queries to the same domain to one per seven seconds. Hence, high crawling speed is only achieved as long as many servers are queued. To increase crawling performance, some basic configurations were considered: In order to avoid link farms and spider traps, we follow links only up to a maximum depth. To reduce download bandwidth we exclude certain kinds of files like images, media files, compressed archives or executable files. Additionally, URLs containing certain keywords (*download*, *files*, *image*, *pics*, *upload*, *redir* or *search*) are excluded from consideration. Further, we restrict the maximum file size to 1 MB to reduce the amount of lists or computer-generated content.

Heritrix creates output files in the Web Archive file format (WARC)[5]. The WARC file format specifies how to combine multiple digital resources with meta-information into a single file for long term archiving or distribution. Further processing steps proposed in this work operate on this representation.

3. Processing and Analyzing Web-Data

Post-processing of harvested web data can be efficiently performed using the *WebCorpus*[6] project. WebCorpus makes use of the highly efficient *Hadoop*[7] framework, which offers the execution of algorithms following the *MapReduce* programming paradigm [6] in a distributed environment. Due to the choice of Hadoop as the basis framework it is possible to process very large data in parallel by a number of computers or just by a single machine. The core idea in MapReduce is to split an algorithm into two phases: *map* and *reduce*. In the map phase, so-called key-value pairs of the input data are produced which are subsequently grouped and combined in the reduce phase by their key to produce the final result. In terms of Hadoop, an algorithm following the MapReduce programming principle is called a *HadoopJob*. The WebCorpus project provides individual HadoopJobs, which are designed to process the data in a pipeline fashion, i.e. one HadoopJob after another.

The general steps of the processing pipeline are described by:

1. **Convert**: converting input data – currently supported input file formats are WARC, ARC[8] and Leipzig Corpora Collection [10] – into a unified document representation, thereby optionally removing html boilerplate text (cf. e.g. [8]),
2. **Filter:** removing duplicate, broken or pointless documents,
3. **Extract:** segmenting, filtering and merging texts in the desired level of granularity – e.g. unique sentences, paragraphs or documents in a particular language,
4. **Annotate:** process texts with UIMA [9] components, e.g. tokenizing, tagging, etc., and parsing
5. **Count:** exploit the resulting annotations by counting n-grams, co-occurrences, subtrees of dependency parses, etc. in the annotated texts.

Figure 1 shows a more detailed overview of the different HadoopJobs in the respective phases. For a description of the individual jobs the reader is referred to [3].

Some of the jobs, in particular the *LanguageJob* and partially also the *SentenceJob* and the *UIMAJob*, are language dependent. For example, the LanguageJob uses the language identification package (JLanI) from the ASV-Toolbox[10] [4], which relies on a precomputed list of high-frequency words for a particular language. These word lists are available for more than 500 languages using mainly Wikipedias, the Universal Declaration of Human Rights and religious texts. More languages could

[5] http://www.digitalpreservation.gov/formats/fdd/fdd000236.shtml

[6] http://sf.net/projects/webcorpus

[7] http://hadoop.apache.org

[8] Internet Archive file format http://www.digitalpreservation.gov/formats/fdd/fdd000235.shtml

[9] http://uima.apache.org

[10] http://wortschatz.uni-leipzig.de/~cbiemann/software/toolbox/

be included on the basis of Bible texts (cf. [5]). Likewise, the *SentenceJob*[11] uses handcrafted sentence breaking rules to segment sentences. While it is equipped with a generic rule set, more specific rules for a particular language will certainly improve the results, but were not considered for the case study in this work for the sake of generality.

Processing 10 GB of web data took around one hour on our compute cluster consisting of eight nodes with eight cores each. The runtime complexity solely depends on the amount of input data and the number of provided machines [1].

The cleaned, language-filtered and preprocessed documents, as well as the various outputs of the count phases like statistically significant co-occurrences or n-grams can then be exploited by a variety of applications, e.g. distributional thesauri or language models (cf. e.g. [2]). In this work, we will exemplify the data with visual analysis of significant co-occurrences using *CoocViewer* [12] [9]. With CoocViewer, co-occurrences of words from multiple arbitrary text corpora can be explored visually in a user-friendly way, providing also access to the source text via full-text indexing. The application itself is divided into two major components:

1. the server-sided data management part, where data is stored in a relational database for fast access through indexes (cf. [10]), and
2. the web based front end, which runs on top of an http server[13]. The browser based client application is thus independent of the underlying operating system and available for many users accordingly.

Screenshots of the application follow in Section 4 where we show the feasibility of processing web-data based on two sample web crawls. As a key characteristic, CoocViewer also comes with the possibility to visualize significant concordances. This feature is particularly useful for analyzing high frequency words.

4. Availability

Corpora as described in the following case study are made available in various ways. On the one hand the full size corpora are accessible online using the web interface of the Leipzig Corpora Collection[14]. On the other hand the textual data can be downloaded[15]. For download corpora of standard sizes of up to 1 million sentences are provided. They can be viewed locally using e.g. the Java Corpus Browser [10]. All textual data underlies creative commons attribution license (cc by)[16] allowing users to use and modify the data freely.

The collected sentences are shuffled such that the original structure of the documents cannot be recovered easily, because of legal issues. This inhibits the reconstruction of the original material. With respect to German copyright legislation this practice is considered legally secured, since there is no copyright on single sentences.

The various pre-processing tools involved in the creation of corpora as described are free to use. Among these are tools for HTML-Stripping, sentence segmentation, sentence cleaning, and language identification[17]. All tools can be utilized for non-commercial users following creative commons attribution-noncommercial license (cc by-nc)[18]. The WebCorpus and the CoocViewer toolkits are available as open-source components in Java under the Apache v2 License[19].

5. Case Study

For our case study, two top-level-domains were crawled, from which we assume that they contain documents of languages that are known to be under-resourced. We tested the .fo domain (Faeroe Islands) and the .ke domain (Kenya), where the languages of interest are Faroese and Kiswahili respectively. Kiswahili is also spoken in other countries such as Tanzania, which could be collected by crawling their respective TLDs. Both domains were crawled using the Heritrix-based crawler, resulting in 1.2 million websites for .fo and 3.1 million websites for .ke. Crawling took about three days for Faroe Islands and four days for Kenya resulting in an average speed of 9 resp. 5 URLs per second. Due to self-imposed politeness restrictions, a maximum download speed of about 200 URLs/s was only reached at the beginning of the crawling process. Higher average rates could easily be achieved by lifting query limits for the cost of being less polite to web server operators.

When conducting language separation, fundamentally different compositions of the domains in question become obvious. More than 60% of the documents of the .fo TLD are written in Faroese, as can be seen in Table 1. English is the second largest language having a 15% share. Next in the ranking are further North Germanic languages, namely Icelandic and Danish.

Table1: Results of language separation using websites of the Faroese domain.

Language	Percentage
Faroese	60.63
English	14.69
Icelandic	11.24
Danish	10.29
French	0.64

[11] The SentenceJob internally uses the ASV-Toolbox.
[12] http://coocviewer.sf.net
[13] Any webserver that supports PHP.
[14] http://corpora.informatik.uni-leipzig.de
[15] http://corpora.informatik.uni-leipzig.de/download.html
[16] https://creativecommons.org/licenses/by/3.0/
[17] http://asvdoku.informatik.uni-leipzig.de/corpora/index.php?id=corpus-pre-processing-tools
[18] http://creativecommons.org/licenses/by-nc/3.0/
[19] http://www.apache.org/licenses/LICENSE-2.0

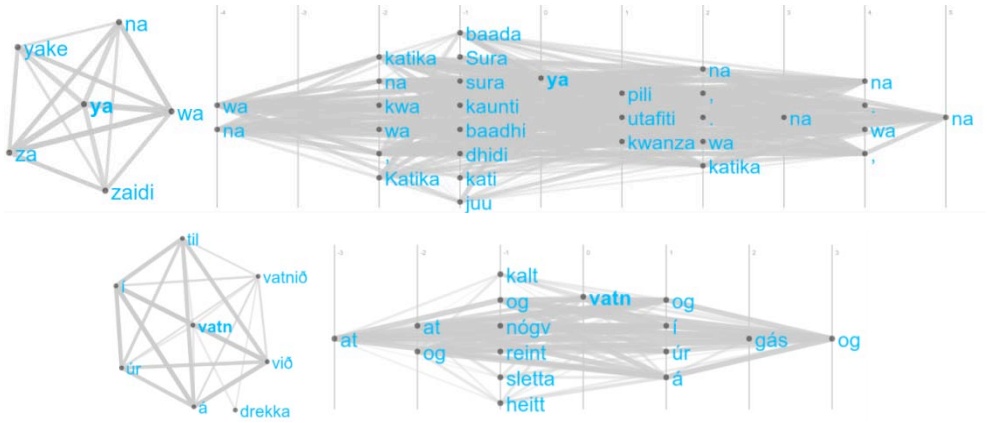


Figure 2: Significant co-occurrences and concordances with the word 'ya' (engl. pendant is roughly 'of', frequency 10,531) from the Kiswahili corpus (upper) and with the word 'vatn' (engl. 'water', frequency 1,504) from the Faroese corpus (lower). Thicker lines represent more significant relations.

When analyzing the Kenyan TLD, an extremely high percentage value for English documents becomes evident (Table 2). Although it is the second largest language among .ke documents, only 0.84% of all texts contain Kiswahili. Together, these two form the national languages of Kenya.

Table2: Results of language separation using websites of the Kenyan domain.

Language	Percentage
English	98.20
Kiswahili	0.84
Russian	0.21
Latin	0.12
Spanish	0.08

The WebCorpus framework was applied as described in Section 3. Only Faroese respectively Kiswahili texts were considered, texts from other languages were filtered in the Filter phase of the WebCorpus pipeline (cf. Sec. 3). Further, we defined our unit of granularity to be the sentence level since our example application is the analysis of co-occurrences of words on a sentence level. After applying the entire WebCorpus pipeline as described in Section 3, we have 7,873 unique sentences and 31,274 types for Kiswahili, and 888,255 unique sentences and 1,030,611 types for Faroese. Yet, the lack of a profound knowledge of these languages makes it impossible for us to judge the quality of the extracted sentences. In particular, the very high number of sentences and tokens for Faroese suggests unclear boundaries in the language separation step. Indeed, during manual inspection of the dataset, we observed some false-positive Danish sentences.

Figure 2 shows the co-occurrences and significant concordances of selected words from either corpus. As should become evident, the setup we described is suited for studies in corpus linguistics and other research.

6. Conclusion

Free and open-source software components have been made available by us and others, that allow researchers and others to produce high quality web corpora targeted to their own needs without relying on the good will of commercial companies to provide it. We have exercised one possible workflow for producing such corpora for two under-resourced languages and conclude, that although we are lacking the needed knowledge for these languages, we are able to produce reasonable results. We assume that further processing of these corpora by experts – mainly cleaning of artefacts from different languages, false segmentation, etc. – would result in high quality corpora from the web. Everybody is thus able to produce web corpora using just the few steps outlined above, and by relying solely on freely available software. By applying some simple configuration settings for Heritrix, the open-source crawler of the Internet Archive, it is easy to crawl specified regions of the World Wide Web in order to collect usable text. By making use of the Hadoop framework, the user herself chooses the level of scalability. Even a single computer is able to run the provided workflow, but when providing more machines, users are able to create corpora of very large sizes in reasonable time.

In two case studies, we have demonstrated how to collect corpora for rather under-resourced languages. Still, the proposed approach can be applied to larger languages if enough computational resources are available. These corpora can form the basis to compute language models, and other NLP components trained from unannotated text.

7. References

[1] Marco Baroni, Silvia Bernardini, Adriano Ferraresi and Eros Zanchetta. 2009. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. *Language Resources and Evaluation* 43 (3): 209-226.

[2] Thorsten Brants, Alex Franz (2006): Web 1T 5-gram

Version 1. *LDC*, Philadelphia, PA, USA

[3] Chris Biemann, Felix Bildhauer, Stefan Evert, Dirk Goldhahn, Uwe Quasthoff, Roland Schäfer, Johannes Simon, Leonard Swiezinski, Torsten Zesch. 2013. Scalable Construction of High-Quality Web Corpora. *Journal for Language Technology and Computational Linguistics (JLCL)*, 28(2), pp. 23–59.

[4] Chris Biemann, Uwe Quasthoff, Gerhard Heyer, and Florian Holz. 2008. ASV Toolbox – A Modular Collection of Language Exploration Tools. In *Proc. of LREC 2008*, Marrakech, Marocco

[5] Michael Cysouw. 2009. Using the World Atlas of Language Structures. In *STUF - Language Typology and Universals Sprachtypologie und Universalienforschung.* 61(3): 181–185. Akademie Verlag, Berlin

[6] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of the 6th Symposium on OSDI*. San Francicso, CA, USA

[7] Adam Kilgarriff. 2007. Googleology is Bad Science. *Computational Linguistics.* 33(1): 147–151.

[8] Christian Kohlschütter, Peter Fankhauser and Wolfgang Nejdl. 2010. Boilerplate Detection using Shallow Text Features. In *Proc. of WSDM,* New York City, NY, USA

[9] Janneke Rauscher, Leonhard Swiezinski, Martin Riedl and Chris Biemann. 2013. Exploring Cities in Crime: Significant Concordance and Co-occurrence in Quantitative Literary Analysis. In *Proc. of the Computational Linguistics for Literature Workshop at NAACL-HLT 2013*, Atlanta, GA, USA

[10] Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, and Chris Biemann. 2006. Exploiting the Leipzig corpora collection. In *Proc. of the IS-LTC.* Ljubljana, Slovenia.

[11] Roland Schäfer and Felix Bildhauer. 2012. Building large corpora from the web using a new efficient tool chain. In *Proc. of LREC 2012*, Istanbul, Turkey

Making a large treebank searchable online. The SoNaR case.

Vincent Vandeghinste, Liesbeth Augustinus
Centre for Computational Linguistics, KU Leuven
Leuven
Email: vincent@ccl.kuleuven.be, liesbeth@ccl.kuleuven.be

Abstract

We describe our efforts to scale up a syntactic search engine from a 1 million word treebank of written Dutch text to a treebank of 500 million words, without increasing the query time by a factor of 500. This is not a trivial task. We have adapted the architecture of the database in order to allow querying the syntactic annotation layer of the SoNaR corpus in reasonable time. We reduce the search space by splitting the data in many small databases, which each link similar syntactic patterns with sentence identifiers. By knowing on which databases we have to apply the XPath query we aim to reduce the query times.

Keywords: corpus indexing, treebanks, search engine

1. Introduction

The research described in this work mainly concerns how to scale up syntactic search from a 1 million word treebank to a 500 million word treebank. In the *Nederbooms* project Augustinus et al. (2012) made the LASSY Small treebank (van Noord et al., 2013) available for online querying using GrETEL.[1] LASSY Small consists of 1 million words which have been syntactically annotated using the Alpino parser (van Noord, 2006), and which have been manually corrected.

The SoNaR corpus (Oostdijk et al., 2013) is a balanced corpus of written Dutch that consists of 500 million words, which are fully automatically tokenized, POS-tagged, lemmatized, and syntactically analysed (van Noord et al., 2013), using the Alpino parser.

Scaling up the treebank from a 1 million word corpus to a 500 million word corpus is not a trivial task. This paper describes the new architecture we had to adopt in order to allow querying the syntactic annotation of the SoNaR corpus in reasonable time.

In section 2 we describe some related work as well as our motivation to use XPath as a query language for the LASSY and SoNaR treebanks. In section 3 we describe the GrETEL 1.0 approach which we use for querying small treebanks. In section 4, we describe scaling up the GrETEL database architecture for very large treebanks. In section 5, the GrETEL 2.0 search engine is presented. Section 6 concludes and describes future work

2. Treebank Querying

Currently there exist many linguistic treebanks and almost as many query languages and treebanking tools to explore those treebanks.

The Penn Treebank (Marcus et al., 1993) should be queried with TGrep2 (Rohde, 2005) via a command-line interface, the Prague Dependency Treebank (Hajič et al., 2006) can be queried through Netgraph (Mírovský, 2008), a client-server application. Several treebanking tools support (extensions of) the TIGER query language (König and Lezius, 2003), such as the standalone tool TIGERSearch (Lezius, 2002), or the online query engines TüNDRA (Martens, 2012; 2013) and INESS-Search (Meurer, 2012). For a comparison of some existing query languages, see Lai and Bird (2004).

In both LASSY Small and SoNaR the syntactic structures are represented in XML format, which can be visualised using the appropriate style sheet.[2] Each XML tree structure is isomorphous to the dependency tree of the syntactic annotation layer. This is not the case for XML representations of syntax trees in Tiger-XML format (König et al., 2003) or FoLiA format (Van Gompel and Reynaert, 2014), in which trees are represented as sets of nodes and links between those nodes. The use of isomorphous XML allows to query for syntactic structures using W3C standard tools such as XPath[3] and XQuery,[4] as explained in van Noord et al. (2013).

DTSearch (Bouma and Kloosterman 2002; 2007) and DACT[5] are two standalone tools for querying syntactic trees in Alpino-XML format using XPath. Since XPath is a standard query language and is already supported in the standalone tools for querying Dutch treebanks, we use it as a query language in GrETEL as well, as explained in more detail in section 3.

1 **Gr**eedy **E**xtraction of **T**rees for **E**mpirical **L**inguistics, http://nederbooms.ccl.kuleuven.be/eng/gretel

2 You can try this at http://www.let.rug.nl/vannoord/bin/alpino or http://nederbooms.ccl.kuleuven.be/eng/alpino.

3 http://www.w3.org/TR/xpath/

4 http://www.w3.org/TR/xquery/

5 http://rug-compling.github.io/dact/

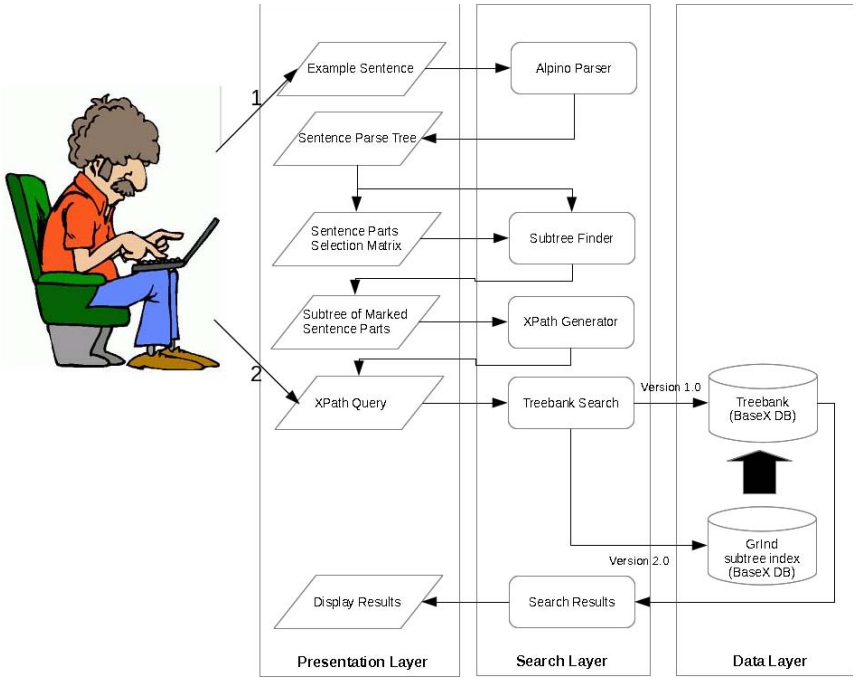


Figure 1: GrETEL architecture

This avoids the common complaint of users having to learn yet another query language. Van Noord et al. (2013) show how the queries presented in Lai and Bird (2004) can be *translated* to the Dutch grammar and fairly easily *converted* into XPath.

In order to support (non-technical) users that are reluctant towards learning any query language at all, we also implemented example-based search, a query system that does not ask for any formal input query. Tools related to that approach are the Linguist's Search Engine (Resnik and Elkiss, 2005), another example-based query engine (which is no longer available) and the TIGER Corpus Navigator (Hellman et al., 2010), a SemanticWeb system that classifies and retrieves corpus sentences based on abstract linguistic concepts.

The system we present here is an online system, which shares the advantages of tools like TüNDRA and INESS-Search: They are platform independent and no local installation of the treebanks is needed. This is especially attractive for (very) large parsed corpora which require a lot of disk space.

3. GrETEL 1.0 (Lassy small)

Figure 1 presents the architecture of the GrETEL search engine. The user has two ways of entering a syntactic query.

The first approach, indicated by (1) in Figure 1, is called *Example-based Querying* (Augustinus et al., 2012; 2013) which consists of a query procedure in several steps. The user provides an example sentence, containing the syntactic construction (s)he is querying. The input sentence is automatically syntactically analysed using the Alpino parser. In the *Sentence Parts Selection Matrix*, the user indicates which part of the entered example (s)he is actually looking for. The *Subtree Finder* extracts the query tree from the full parse tree. The *XPath Generator* automatically converts the query tree into an XPath expression. The XPath query is then used to search the treebank stored in the BaseX database system (Holupirek and Scholl, 2008), which is a native XML database system optimised for XQuery and XPath performance.[6] For example, if one is looking for constructions containing the adverbial phrase *lang niet altijd* "not always", a possible input example is the construction *Het is lang niet altijd gemakkelijk* "It is far from easy". After indicating the relevant parts of the input construction, GrETEL extracts the subtree in Figure 2, and turns it into the XPath expression in Figure 3.

6 http://basex.org/

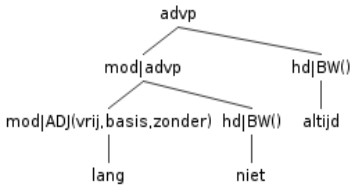


Figure 2: An example bottom-up subtree.

```
//node[@cat="advp" and node[@rel="mod"
and @cat="advp" and node[@rel="mod" and
@pt="bw" and @lemma="lang"] and
node[@rel="hd" and  @pt="bw" and
@lemma="niet"]] and node[@rel="hd" and
@pt="bw" and @lemma="altijd"]]
```

Figure 3: XPath based on Figure 2

Example-based querying has the advantage that the user does not need to be familiar with XPath, nor with the exact syntactic sugar of the XML in which the trees are represented, nor with the exact grammar implementation that is used by the parser or the annotators. Nevertheless, XPath querying greatly enhances the query flexibility compared to the example-based approach.

Therefore, the second approach, indicated by (2) in Figure 1, consists of directly formulating an XPath query describing the syntactic pattern the user is looking for. This query is then processed in the same way as the automatically generated query in the first approach.
The online query engine is fast, but if one is looking for rare constructions, little or no results are found since the size of the treebank is rather small. We want to overcome this problem by including the large SoNaR corpus, but this introduces another challenge: Making such a large treebank searchable in *reasonable*-time, in order to implement it for online querying.

4. GrInding the data per breadth-first pattern

The general idea behind our approach is to restrict the search space by splitting up the data in many small databases, allowing for faster retrieval of syntactic structures. We organise the data in databases that contain all bottom-up subtrees for which the two top levels (i.e. the root and its children) adhere to the same syntactic pattern.
When querying the database for certain syntactic constructions, we know on which databases we have to apply the XPath query which would otherwise have to be applied on the whole data set. We have called this method GrETEL Indexing (GrInd).
As an example, take the parse tree presented in Figure 4.

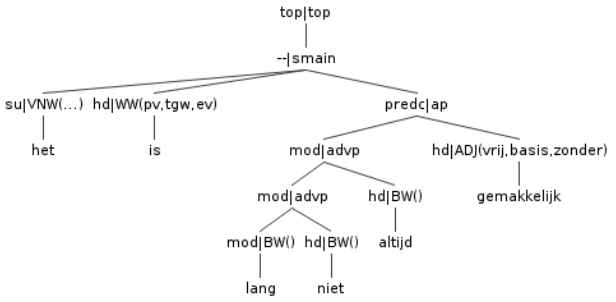


Figure 4: An example parse tree

In a top-down fashion, we take for each node all the bottom-up subtrees, i.e. all the subtrees that have only lexical elements as their terminals.[7] For instance, taking the example parse tree from Figure 4, for the `--|smain` node, we extracted the subtree with three daughters shown in Figure 5a, with two daughters in Figure 5b and with only one daughter in Figure 5c.
This procedure is applied recursively for each node in the parse tree. For every extracted subtree, we convert the children of the root into a string-based breadth-first pattern, inspired by Chi et al. (2005) and taken over from Vandeghinste and Martens (2010).[8] As the trees are dependency trees, the order of children is not important. Therefore, the children in the breadth-first representation are sorted in a canonical alphabetical order. Note also that the POS tags included in the trees contain more detailed information such as gender, number and agreement. In the breadth-first strings, only the general POS tag is used. For the subtrees in Figure 5, this amounts to the following breadth-first patterns respectively:

```
hd%ww_predc%ap_su%vnw
hd%ww_su%vnw
su%vnw_predc%ap
predc%ap_hd%ww
su%vnw
hd%ww
predc%ap
```

Those breadth-first patterns are combined with the category label of their root (in this case `smain`). To the file with that name we copy the XML of the subtree, adding the sentence identifiers indicating where these subtrees come from.
We organise these breadth-first files per corpus component.[9]

7 This definition differs from the definition of bottom-up subtrees used in Vandeghinste et al. (2013) in the sense that in this case, they do not have to be horizontally complete.
8 Opposed to Vandeghinste and Martens (2010) only the children of the root are converted into breadth-first patterns.
9 SoNaR contains 25 components, each containing a different text genre.

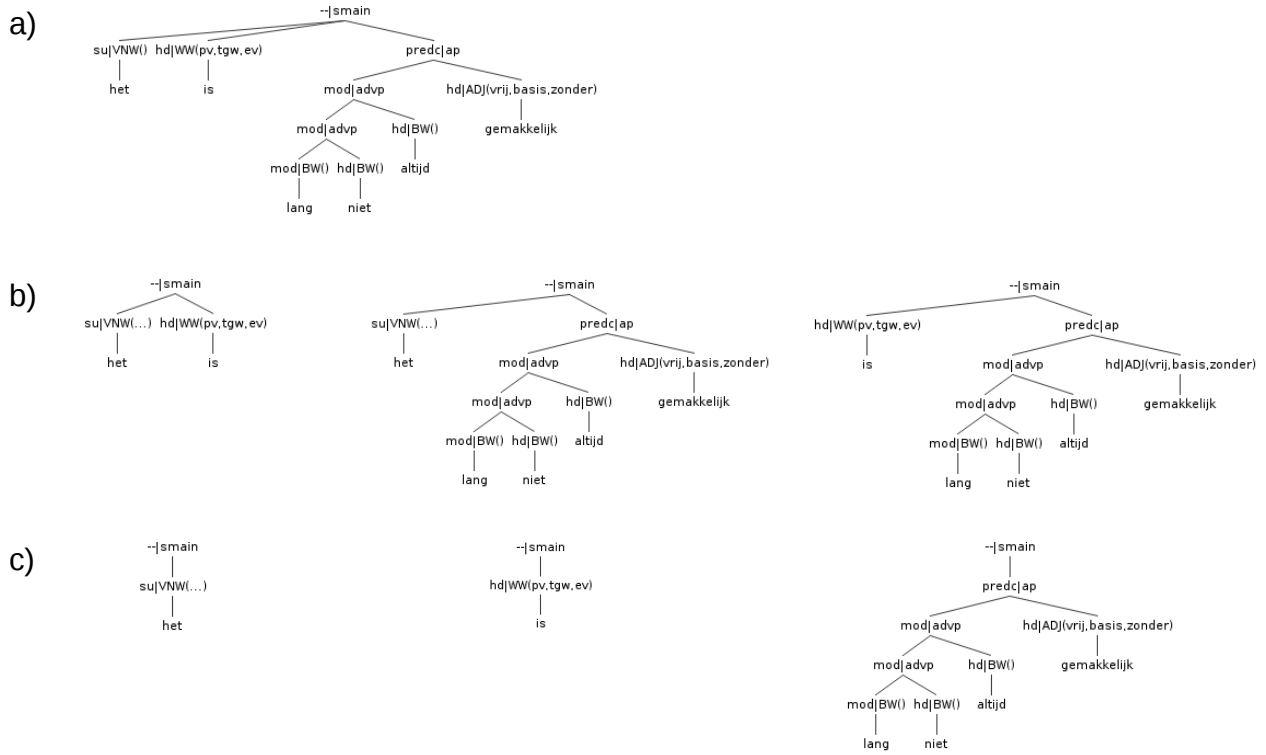


Figure 5: Bottom-up subtrees of the `smain` node in Figure 4

For example, in the WR-P-E-F component[10] of SoNaR, we have extracted all adverbial phrases that have as children an adverbial phrase (advp) as a modifier (mod) and an adverb (bw) as a head (hd), and put them in the following XML file `WRPEFadvphd%bw_mod%advp.xml`. The XML structure in Figure 6 corresponds to the subtree in Figure 2.

```
<treebank component="WRPEF" cat="advp" file="hd
%bw_mod%advp">
 <tree id="WR-P-E-F-0000000769.p.4.s.7" >
  <node begin="3" cat="advp" end="6" id="6"
rel="mod">
   <node begin="3" cat="advp" end="5" id="7"
rel="mod">
    <node begin="3" buiging="zonder" end="4"
frame="adverb" graad="basis" id="8" lcat="advp"
lemma="lang" pos="adv" positie="vrij"
postag="ADJ(vrij,basis,zonder)" pt="adj"
rel="mod" root="lang" sense="lang" word="lang"/>
    <node begin="4" end="5" frame="adverb" id="9"
lcat="advp" lemma="niet" pos="adv" postag="BW()"
pt="bw" rel="hd" root="niet" sense="niet"
word="niet"/>
   </node>
   <node begin="5" end="6" frame="adverb" id="10"
lcat="advp" lemma="altijd" pos="adv"
postag="BW()" pt="bw" rel="hd" root="altijd"
sense="altijd" word="altijd"/>
  </node>
 </tree>
</treebank>
```

Figure 6: XML containing a subtree

By extracting all the bottom-up subtrees from a given parse tree, and copying them to the files with the appropriate breadth-first filename, higher-level subtrees contain copies of lower level subtrees, and the size of the data grows considerably. In order to avoid copying the information from horizontally complete subtree patterns to the horizontally incomplete variants, we use `<include>` tags, indicating in which other files the queried pattern might occur as well. More general patterns are included in more specific patterns.

For instance, when looking for adverbial phrases (advp) that have a modifying adverbial phrase (mod|advp) as daughter, one should also look for this pattern in the file that contains the adverbial phrases that have a modifying adverbial phrase AND a head adverb (hd|bw) as daughters, resulting in a file `WRPEF/advp/bfmod%advp.xml` with the following data:

```
<treebank component="WRPEF" cat="advp" file="mod
%advp">
  <include file="WRPEFadvpmod%advp_hd%bw" />
</treebank>
```

Figure 7: XML of a subtree that is included in another database

When a subtree is found that matches the XPath query, we retrieve the sentence identifier, as indicated by the `id` feature in the `<tree>` tag, as shown in Figure 6, and

10 The WR-P-E-F component contains press releases.

display the full sentence and optionally the full parse tree as retrieved from the original treebank.

5. Querying the data (GrETEL 2.0)

Similar to the query engine for Lassy small, we use BaseX (Holupirek and Scholl, 2008), a native XML database system, as an XPath query engine for SoNaR. We have created over 10 million databases in BaseX, each with a name that is indicative of the components and patterns that are described therein. GrInd allows us to query SoNaR in a faster and more efficient way, whereas earlier attempts to query the corpus resulted in memory issues.

However, some queries still take too long for online searching, which caused a browser time-out. Therefore, the current system outputs a sample of the results and their frequency during the first *n* seconds of querying.[11] By caching the queries and the results we avoid querying the same construction multiple times, and it allows us to return the final results immediately.

In the next version we plan to implement a messaging system which notifies when the user can find the results once the search action is completed.

As described in section 3 we have two ways of feeding queries to the syntactic search engine: Through an example and through an XPath expression. The first version of the online search engine for SoNaR only allows the *example-based querying* method.

When we are querying via an example (query method 1 in Figure 1), the search engine extracts a bottom-up query subtree from the parse tree of the (natural language) input example. When we are querying the treebank looking for this bottom-up subtree, we know in which BaseX database we have to look, as we can construct the name of the database based on the query subtree. As described in the previous section, the database name depends on the syntactic category of the root and on the syntactic categories and dependency relations of the children of the root. It is only in trees in this database that the query subtree can occur, and therefore we do not need to look in the rest of the treebank. The query subtree is also automatically converted into XPath, and it is this XPath expression that is used as a query in the relevant databases.

When we are querying with an XPath expression (query method 2 in Figure 1), there is no subtree that can be used to retrieve the relevant databases. Instead, we have to extract the largest bottom-up subtree that complies with the XPath expression in order to determine where to look for the relevant patterns. This is not a trivial process, as XPath can contain conjunctions (AND), optionality (OR), and negation (NOT), defining multiple query trees in one query, not necessarily with the same root category label and the same children of the root, and hence not necessarily in the same database. Solving this issue remains future work.

11 Currently, *n* is set to 60 seconds, but parametrisable.

6. Conclusions and future work

We have described how we have organised the data of a very large treebank of written Dutch in order to search syntactic constructions within reasonable query times. When the user is interested in frequencies of syntactic patterns, query times will be longer, as all the relevant databases from all the relevant components need to be queried. Nevertheless we are not aware of any other approach towards large treebank querying that allows faster querying.

The work described in this paper is still in progress. We still have to implement extracting the largest bottom-up tree complying with an XPath expression, which is a non-trivial task, as XPath expressions can contain optionality and negation. As already mentioned, we will include a messaging system to work around excessive query times, which we still occasionally expect. Finally, we also plan to apply this architecture to treebanks for Afrikaans and English, increasing the coverage of the GrETEL search engine.

7. Acknowledgements

The research presented in this paper is part of GrETEL 2.0, a project on the adaption of the GrETEL search engine for querying very large treebanks, sponsored by CLARIN-NTU.

8. References

Liesbeth Augustinus, Vincent Vandeghinste, Ineke Schuurman, and Frank Van Eynde (2013). Example-Based Treebank Querying with GrETEL – now also for Spoken Dutch. In *Proceedings of the 19th Nordic Conference on Computational Linguistics (NoDaLiDa 2013).* NEALT Proceedings Series 16. Oslo, Norway. pp. 423-428.

Liesbeth Augustinus, Vincent Vandeghinste, and Frank Van Eynde (2012). Example-Based Treebank Querying. In *Proceedings of LREC 2012.* Istanbul, Turkey. pp. 3161-3167

Gosse Bouma and Geert Kloosterman (2002). Querying Dependency Treebanks in XML. In *Proceedings of LREC'02.* Las Palmas, Spain. pp. 1686–1691.

Gosse Bouma and Geert Kloosterman (2007). Mining Syntactically Annotated Corpora with XQuery. In P*roceedings of the Linguistic Annotation Workshop.* Prague, Czech Republic. pp. 17–24.

Yun Chi, Siegfried Nijssen, Richard Muntz, and Joost Kok (2005). Frequent Subtree Mining An Overview. *Fundamental Informatics, Special Issue on Graph and Tree Mining.* pp. 1001-1038.

Jan Hajič, Jarmila Panevová, Eva Hajičová, Petr Sgall, Petr Pajas, Jan Štěpánek, Jiří Havelka, Marie Mikulová, Zdeněk Žabokrtský and Magda Ševčíková-Razímová

(2006). Prague Dependency Treebank 2.0. CD-ROM LDC2006T01, LDC, Philadelphia.

Sebastian Hellmann, Jörg Unbehauen, Christian Chiarcos, and Axel-Cyrille Ngonga Ngomo (2010). The TIGER Corpus Navigator. In *Proceedings of the The Ninth Workshop on Treebanks and Linguistic Theories (TLT9)*. Tartu, Estonia. pp. 91-102.

Alexander Holupirek and Marc H. Scholl (2008). An XML Database as Filesystem in Userspace. In *Proceedings of the 20. GI Workshop on Foundations of Databases, August 2008*. School of Information Technology, Germany. pp. 31-35.

Ester König and Wolfgang Lezius (2003). The TIGER language - A Description Language for Syntax Graphs, Formal Definition. Technical report. IMS, University of Stuttgart, Germany.

Esther König, Wolfgang Lezius, and Holger Voormann (2003). TIGERSearch User's Manual. IMS, University of Stuttgart, Germany.

Catherine Lai and Steven Bird (2004). Querying and updating treebanks: a critical survey and requirements analysis. In *Proceedings of the Australasian Language Technology Workshop*. Sydney, Australia. pp. 139–146.

Wolfgang Lezius (2002). TIGERSearch: ein Suchwerkzeug für Baumbanken. In *Proceedings of KONVENS-02*. Saarbrücken, Germany.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Scott Martens (2012). TüNDRA: TIGERSearch-style treebank querying as an XQuery-based web service. In *Proceedings of the joint CLARIN-D/DARIAH Workshop "Service-oriented Architectures (SOAs) for the Humanities: Solutions and Impacts" (DH 2012)*. Hamburg. pp. 41-50.

Scott Martens (2013). TüNDRA: A Web Application for Treebank Search and Visualization. In *Proceedingsof the The Twelfth Workshop on Treebanks and Linguistic Theories (TLT12)*. Sofia, Bulgaria. pp. 133-143.

Paul Meurer (2012). INESS-Search: A search system for LFG (and other) treebanks. In *Proceedings of the LFG'12 Conference*. LFG Online Proceedings. Stanford, CSLI Publications. pp. 404–421

Jiří Mírovský (2008). Netgraph Query Language for the Prague Dependency Treebank 2.0. *The Prague Bulletin of Mathematical Linguistics No. 90*. pp. 5–32.

Nelleke Oostdijk, Martin Reynaert, Véronique Hoste, and Ineke Schuurman (2013). The Construction of a 500-million-word Reference Corpus of Contemporary Written Dutch. In Peter Spyns and Jan Odijk (eds.): *Essential Speech and Language Technology for Dutch: resources, tools and applications.* Springer.

Philip Resnik and Aaron Elkiss (2005). The Linguist's Search Engine: An Overview. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions.* Ann Arbor. pp. 33-36.

Douglas L.T. Rohde (2005). TGrep2 User Manual.

Vincent Vandeghinste, Scott Martens, Gideon Kotzé, Jörg Tiedemann, Joachim Van Den Bogaert, Koen De Smet, Frank Van Eynde, and Gertjan van Noord (2013). Parse and Corpus-based Machine Translation. In Peter Spyns & Jan Odijk (eds.): *Essential Speech and Language Technology for Dutch: resources, tools and applications.* Springer.

Vincent Vandeghinste and Scott Martens (2010). Bottom-up transfer in Example-based Machine Translation. In François Ivon and Viggo Hansen (eds.) In *Proceedings of the 14th International Conference of the European Association for Machine Translation (EAMT-2010).* Saint-Raphael, France.

Maarten van Gompel and Martin Reynaert (2014). FoLiA: A practical XML format for linguistic annotation - a descriptive and comparative study. *Computational Linguistics in the Netherlands Journal* 3:63-81.

Gertjan van Noord. 2006. At Last Parsing Is Now Operational. In *TALN 2006*. pp. 20–42.

Gertjan van Noord, Gosse Bouma, Frank Van Eynde, Daniel de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste (2013). Large Scale Syntactic Annotation of Written Dutch: Lassy. In: Peter Spyns and Jan Odijk (eds.): *Essential Speech and Language Technology for Dutch: Resources, Tools and Applications.* Springer.

Dealing With Big Data Outside Of The Cloud: GPU Accelerated Sort

John Vidler*, Paul Rayson*, Laurence Anthony†, Andrew Scott*, John Mariani*
*School of Computing and Communications, Lancaster University
j.vidler, p.rayson, a.scott, j.mariani@lancaster.ac.uk
†Faculty of Science and Engineering, Waseda University
anthony@waseda.jp

Abstract

The demands placed on systems to analyse corpus data increase with input size, and the traditional approaches to processing this data are increasingly having impractical run-times. We show that the use of desktop GPUs presents a significant opportunity to accelerate a number of stages in the normal corpus analysis pipeline. This paper contains our exploratory work and findings into applying high-performance computing technology and methods to the problem of sorting large numbers of concordance lines.

Keywords: Very Large Corpora, Concurrency, GPU Computing, High Performance Computing, Concordances, Sorting

1. Introduction

Corpus data is used in many areas of Digital Humanities, Natural Language Processing, Human Language Technologies, Historical Text Mining and Corpus Linguistics. Increasingly, however, the size of corpus data is becoming unmanageable. In Digital Humanities, for example, national and international digitisation initiatives are bringing huge quantities of archive material in image and full text form direct to the historian's desktop. Processing such data quickly, on the other hand, will almost certainly exceed the limits of traditional database models and desktop software packages. Similarly, the "Web as a Corpus" paradigm has brought vast quantities of Internet-based data to corpus linguists. However, any search or sort of results from these rich datasets is likely to take from minutes to hours to days using desktop corpus tools such as WordSmith Tools[1] and AntConc[2].

To address the problems of handling massive data sets, international infrastructure projects, such as CLARIN and DARIAH, are emerging with support for these large corpora under the umbrella of 'big data'. However, these systems do not allow for local access, storage and retrieval of large language resources to support researchers while datasets are being collected and analysed. In corpus linguistics, researchers now have access to tools such as Sketch Engine[3] and the family of BYU Corpora[4], which aim to support pre-compiled billion-word corpora. Again, though, these systems are remotely hosted, and they are also not easy to configure for the new datasets of local researchers. More recently, semi-cloud based systems are emerging, such as GATE[5], Wmatrix[6], and CQPweb[7], which can provide users with local access to large data sources. However, the installation and configuration of such systems is far from simple, making them inaccessible to most social science and humanities based scholars.

Hence, there is still a need to investigate processing efficiency improvements for locally controlled and installed corpus retrieval software tools and databases. Core tasks such as corpus indexing, calculating n-grams, creating collocations, and sorting results on billion-word databases cannot feasibly be carried out on current desktop computers within a reasonable time.

In this paper, we describe an alternative solution to accelerate such tasks by capitalizing on the local processing power of the often underused discrete Graphics Processing Unit (GPU). To highlight the possibilities of our approach, we focus on the task of concordance results sorting and show how GPU hardware can dramatically shorten the time needed to complete the task. This research forms our first case study in a larger project to investigate the untapped potential in current operating system architecture designs.

2. Background

As corpus sizes increase, the problems of processing large datasets are become more pressing. Research on high performance processing techniques for the amounts of text that the language resources community often works with is scant at best, and generally revolves around large numbers of traditional processors being used to divide the work into manageable units. Unfortunately, with corpora exceeding the multi-billion-word mark, even these measures are unable to complete experiments within reasonable time, often spanning days of operation (Wattam et al., 2014). In addition, enhancements designed for other areas of computing, e.g., Cederman and Tsigas (2010) and Rashid et al. (2010) have proved to be not well suited to corpus processing. In recent years, high-performance, general-purpose graphics processing units have become increasingly available to the scientific community, and projects utilising them have been met with considerable success as described in Deng (2010), Melchor et al. (2008) and Sun et al. (2012). On the other hand, their use in corpus linguistics and natural language processing has been limited at best, and many areas of their uses have yet to be explored.

[1] http://www.lexically.net/wordsmith/
[2] http://www.antlab.sci.waseda.ac.jp/software.html
[3] http://www.sketchengine.co.uk/
[4] http://corpus.byu.edu/
[5] http://gate.ac.uk/
[6] http://ucrel.lancs.ac.uk/wmatrix/
[7] http://cqpweb.lancs.ac.uk/

the crowds of inquisitive people began to diminish and soon	there	were no more visitors Madame Caravan returning to her own
already to regard the corpse as though it had been	there	for months He even went the length of declaring that
He even went the length of declaring that as yet	there	were no signs of decomposition making this remark just at
the girl who had ascended the stairs were distinctly heard	There	was silence for a few seconds and then the child
in their places and was ready to go downstairs when	there	appeared before her her son and daughter-in-law Caravan rushed forward
agree to be bound by the terms of this agreement	There	are a few things that you can do with most
full terms of this agreement See paragraph C below	There	are a lot of things you can do with Project

Figure 1: A sample of the input set used for testing the sorting algorithms.

3. Method

To evaluate the potential gains of using General Purpose Graphics Processing unit (GPGPU) techniques for corpus retrieval operations, we chose to focus on one of the most common and time consuming tasks that corpus linguists need to perform, i.e., sorting the concordance lines generated from a database query. Once concordance lines are extracted or displayed in a corpus retrieval system, corpus linguists need to identify language patterns in the results set. Due to the large number of results, the concordance lines cannot be skim read manually, and so some preprocessing is required, typically sorting. Most concordancing tools, such as WordSmith Tools and AntConc, can perform a multi-level sort of the results based on the preceding and/or following words. However, this can be a very lengthy operation, especially when many hundreds of thousands of concordance lines emerging from large corpora require processing.

The words we targeted for the corpus sort experiment were taken from the published BNC frequency lists of "Word Frequencies in Written and Spoken English" (Rayson et. al)[8], which were used with a corpus generated from the Gutenburg Project books data[9] to generate CSV input sets as shown in Figure 1. These were loaded into memory in full, and stored such that the entire concordance was kept in RAM. While this may seem suboptimal, we did this to attempt to present data that represented the performance of the GPGPU device, rather than memory usage tricks. Additionally, the batch-processing style of operation used in GPGPU computing limits our ability to do most traditional sorting techniques for large datasets, such as an external merge sort, as the GPU does not support the recursion depth required to process this data. Following the above procedure, we could reduce the problem to an entirely data oriented issue and avoid the characteristics of disks, buses, networks and other hardware components interfering with the performance measures.

Based on a preliminary investigation of different sorting algorithms, we found most to be data-copy sensitive (requiring many short batch operations and many host-device memory copies). Thus, we settled on using the simplest sorting technique, the swapping sort, for the analysis here. In the swapping sort, concordance lines are directly loaded into memory on the graphics card and processed in place by comparing each line to its immediate neighbours in the input set, and swapping the entries if they are incorrectly ordered. This is repeated until the entire set is sorted. While this technique would be extremely inefficient on a CPU, it works impressively well on a GPU, as we can perform large batches (over 27,000 entries, on a nVidia GTX Titan) at once. The relevant specifications of the machine used for these tests is described in Figure 2.

- CPU: Intel "Sandy Bridge" i7, desktop edition (quad core with hyperthreading support).
- GPU: nVidia "GTX Titan" graphics card, 6GB video memory.
- RAM: 6GB Triple Channel memory.
- Disk: A generic 64GB 6GB/s SSD

Figure 2: The specification of the machine used to perform the tests.

While the exact specification for the hardware is not particularly critical, to achieve similar results, the use of a GTX Titan or better is recommended, as older cards have smaller video memory areas, resulting in higher instances of copying to and from the hard drive. The SSD is not required, but was used for these tests to expediate the test duration through eliminating the delays normally incurred through using mechanical disks.

4. Results

The results of our tests can be seen in Figures 3 and 4[10]. Each sort phase used up to 10 words to the right of the selected collocation word to sort the concordance against its neighbours. Phases were repeated until a phase resulted in no swap operations, and thus, the set was completely sorted.

As can be seen in Figures 2 and 3, the GPU accelerated sort consistently beats the sort on a normal CPU except for very small input sets. Below an input set size of 2000 concordance lines, the CPU has a slight advantage, as the GPU has a small delay involved with deploying CUDA kernels[11], causing the overall throughput to dip. On the other hand, beyond 2000 concordance lines, the GPU is several orders of magnitude faster than the CPU, and remains consistently better throughout.

5. Discussion and Conclusions

The results here show that normal CPU processing becomes impractical when sorting beyond 40,000 concordance lines.

[8] http://ucrel.lancs.ac.uk/bncfreq/flists.html

[9] http://www.gutenberg.org/wiki/Gutenberg:The_CD_and_DVD_Project

[10] The data for these plots as well as additional data can be found at http://johnvidler.co.uk/academia/cmlc-2014/

[11] A CUDA kernel is the GPGPU equivalent of a CPU thread

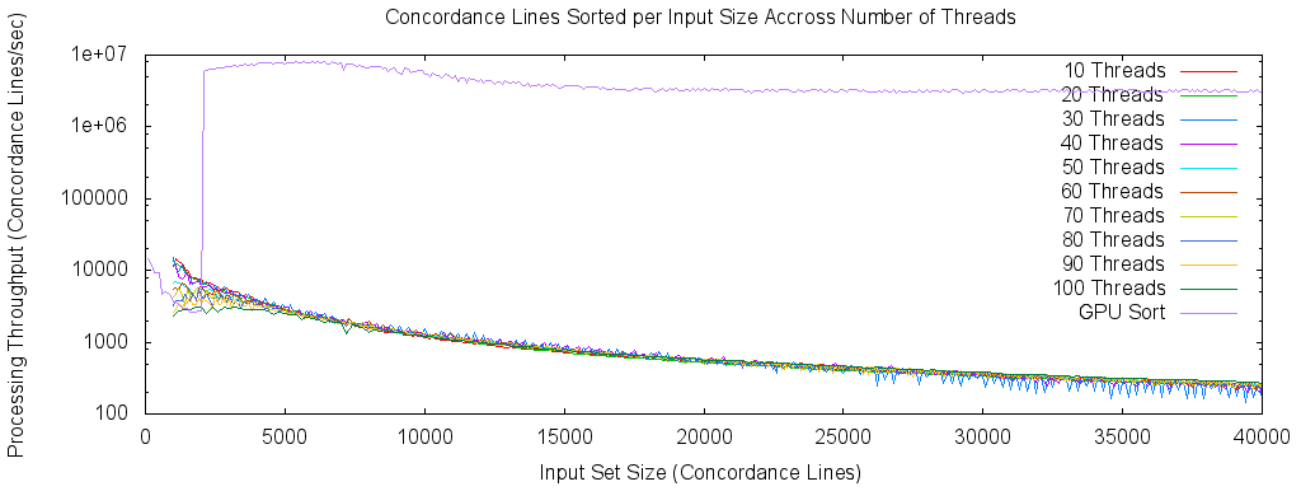


Figure 3: The measured CPU and GPU performance measurements shown on the same axis. Beyond 40,000 concordance lines, the sorting technique took so long to complete on the CPU as to be useless, while the GPU continued to perform exceptionally well. Note that the y-scale is logarithmic.

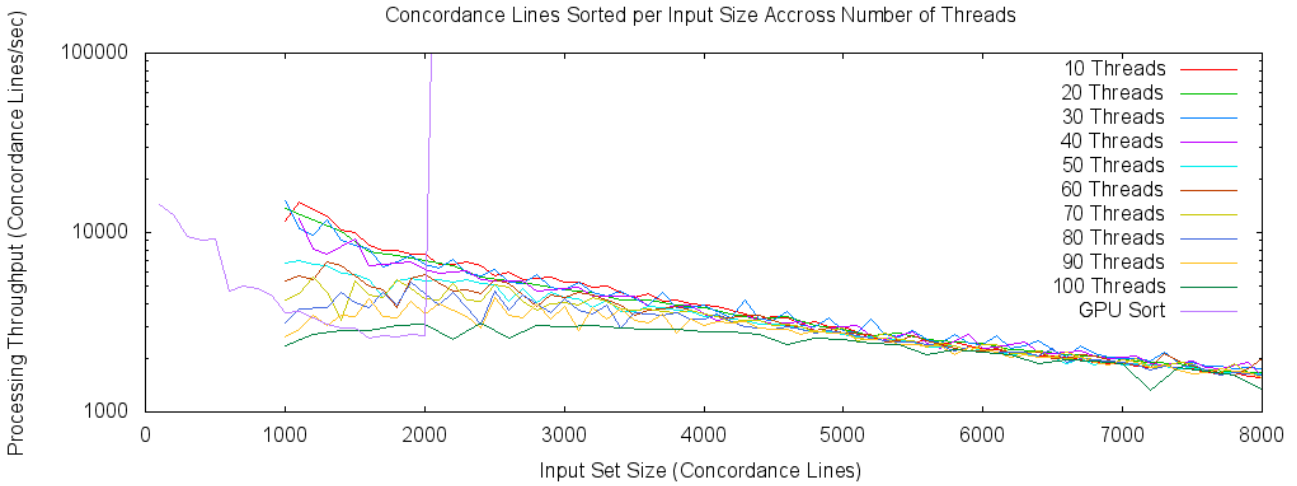


Figure 4: The first portion of Figure 3, showing the initial CPU advantage for very small numbers of concordance lines.

In contrast, our framework for GPU processing allows such operations to be completed exceedingly quickly, e.g., with 10 million concordance lines being processed per second even with large datasets. Further gains can be made through the use of threading on GPGPU devices, although the library support available to the developer is not what one would expect. This leads to problems implementing more traditional sorting algorithms, such as external merge sorting, with such a large input set. Many traditional sorting algorithms require access to the hard drive and other resources that are unavailable to the card during runtime. The processing used in our tests is best described as 'out-of-data-path' processing, as the data would not naturally be processed where we are processing it. Normally one would expect that moving the data further from where it is stored would result in slower overall performance. However, with the immense processing power available in a GPU, once there, the gains more than make up for the longer data path. Of course, while out-of-data-path processing may not be ideal from a memory utilization perspective, our results show that the performance benefits are worth the additional overhead in the application described here and indeed, our approach is likely to be useful in many other areas of natural language processing. In conclusion, our results show that the implementation of even simple algorithms on GPU hardware has significant promise for linguistic analysis of large corpora. Our approach thus has important implications for the development of more powerful desktop linguistic analysis tools. Given the performance shown in the case study presented in this paper, we next intend to implement other standard corpus retrieval operations using our framework. Following this, we will start implementing support tools for various other corpus annotation and NLP operations, e.g. Part-Of-Speech (POS) tagging.

References

Daniel Cederman and Philippas Tsigas. Gpu-quicksort: A practical quicksort algorithm for graphics processors. *J. Exp. Algorithmics*, 14:4:1.4–4:1.24, January 2010. ISSN 1084-6654. doi: 10.1145/1498698.1564500. URL http://doi.acm.org/10.1145/1498698.1564500.

Yangdong Steve Deng. IP routing processing with graphic processors. *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pages 93–98, March 2010. doi: 10.1109/DATE.2010.5457229. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5457229.

Carlos Aguilar Melchor, Benoit Crespin, Philippe Gaborit, Vincent Jolivet, and Pierre Rousseau. High-Speed Private Information Retrieval Computation on GPU. In *Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*, pages 263–272, Washington, DC, USA, August 2008. IEEE Computer Society. ISBN 978-0-7695-3329-2. doi: 10.1109/SECURWARE.2008.55. URL http://portal.acm.org/citation.cfm?id=1447563.1447928.

Layali Rashid, WessamM. Hassanein, and MoustafaA. Hammad. Analyzing and enhancing the parallel sort operation on multithreaded architectures. *The Journal of Supercomputing*, 53(2):293–312, 2010. ISSN 0920-8542. doi: 10.1007/s11227-009-0294-5. URL http://dx.doi.org/10.1007/s11227-009-0294-5.

Weibin Sun, Robert Ricci, and Matthew L. Curry. GPUstore. In *Proceedings of the 5th Annual International Systems and Storage Conference on - SYSTOR '12*, pages 1–12, New York, New York, USA, 2012. ACM Press. ISBN 9781450314480. doi: 10.1145/2367589.2367595. URL http://dl.acm.org/citation.cfm?id=2367595.

Stephen Wattam, Paul Rayson, Marc Alexander, and Jean Anderson. Experiences with Parallelisation of an Existing NLP Pipeline : Tagging Hansard. In *Proceedings of The 9th edition of the Language Resources and Evaluation Conference*, 2014.

From several hundred million to some billion words: Scaling up a corpus indexer and a search engine with MapReduce

Jordi Porta
Departamento de Tecnología y Sistemas
Centro de Estudios de la Real Academia Española
c/ Serrano 187-189, Madrid 28002. Spain
porta@rae.es

Abstract

The growing size of corpora poses some technological challenges to their management. To reduce some of the problems arising in processing a few billion (10^9) words corpora, a shared-memory multithreaded version of MapReduce has been introduced into a corpus backend. Results on indexing very large corpora and computing basic statistics in this parallel processing framework on multicore computers are reported.

Keywords: Corpus Indexing, Corpus Search Engine, MapReduce, Very Large Corpora Management.

1. Introduction

Corpus platforms to analyse very large corpora are scientific instruments that create new opportunities not only for linguistic research but also for research in all text-centric humanities. Basic common operations needed in corpus exploitation are the creation of sub-corpora, the production of lists or concordances of linguistic elements, collocation extraction and distributional analysis. Corpus search engines should give support to many types of queries like frequency counts, distributions, collocations or n-grams in subsets or over corpus partitions.

Corpora size has experienced an upward progression since its beginnings in terms both of primary and annotation data. In the 1960s and 1970s, the first generation of corpora, as the Brown or the LOB corpora, reached one million words each. In the 1990s, bigger corpora were released, as the British National Corpus (BNC), a 100 million word corpus, or the Bank of English (BoE), with 320 million words. The compilation of corpora for other languages was an active area in that decade. At the end of the 2000s, corpora became bigger and diverse. Examples of corpora at that time are the Corpus of Historical American English (COHA) and the Corpus of Contemporary American English (COCA), containing 400 and 450 million words respectively (Davies, 2009; Davies, 2010). At the time of writing this paper, many corpora reach sizes between five and ten thousand million words (Jakubíček et al., 2013; Kupietz et al., 2010). A unique case is Google, which has released its n-grams corpus, obtained from 5.2 million books with OCR techniques, totalling 500 thousand million words.

The increasing size of corpora poses some technological challenges to their management. We have introduced MapReduce into a corpus backend with the aim of indexing very large corpora and integrating online dynamic statistical computations within the query engine. In this paper, we report some of the results of the integration of shared-memory MapReduce into a corpus backend running on multicore processors.

2. The backend for corpus management

Roughly speaking, backends for corpus management could be classified into two families: those based in relational databases (Davies, 2005; Schneider, 2012), and those based in inverted files, as Poliqarp (Janus and Przepiórkowski, 2007), Manatee (Rychlý, 2000; Rychlý, 2007), or KorAP (Schnober, 2012), having CQP as a model system (Christ, 1994).

The system we are presenting here belongs to the same family of CQP. It has been in use since year 2000 as an indexing and searching backend giving in-house support to lexicographers and linguists in exploiting some annotated corpora —CREA, CORDE, CORPES, CDH and others, including combinations[1]— reaching a maximum corpus size of 500 million words. An earlier lexicometric exploitation example of this backend is given in Porta and Ruiz-Ureña (2003). The search engine is also currently answering, on average, no less than two million look-ups per day in some online dictionaries: *Diccionario de la lengua española* (DRAE)[2], *Diccionario panhispánico de dudas* (DPD)[3], *Diccionario esencial*[4] and *Diccionario de americanismos*[5], and in the electronic edition of reference works such as the *Nueva grámatica de la lengua española*[6] or the *Ortografía de la lengua española*[7].

As for ISO:24612 (2012), a corpus is seen as a sequence of textual positions where information in the form of attribute-value pairs can be assigned. Wordforms, lemmas or morphosyntactic descriptions are usually associated to corpus positions but, depending on the application, many other attributes can also be assigned: modern wordforms, part-of-speech abstractions, combinations of morphological features, syntactic function labels, and so on. Additionally, positional attributes can be multivalued in order to cope with

[1] http://www.rae.es/recursos/banco-de-datos
[2] http://lema.rae.es/drae
[3] http://lema.rae.es/dpd
[4] http://lema.rae.es/desen
[5] http://lema.rae.es/damer
[6] http://aplica.rae.es/grweb
[7] http://aplica.rae.es/orweb

ambiguity in annotation. Any XML markup appearing in primary data is considered part of the data stream, but it will not be considered part of the sequence of linguistic elements at the level of tokens. Document's meta information like country, year, genre, author, etc. is encoded as attributes of the document's root element and is used to restrict searches as well as to compute frequency distributions and dispersion.

Indexing is a key point for sophisticated corpus search and retrieval. Positional or text attributes are indexed using inverted files, which allow the linear time processing of queries combining restrictions on attribute-value pairs, as well as distances between positions or other restrictions on short sequences (Zobel and Moffat, 2006). XML markup indexing and querying has also been implemented in the backend: XML elements and element's attribute-value pairs are indexed as regions represented by its document offset and length as well as by the identifiers of the first and the last token of the region they cover. Structural indexes are implemented with interval trees on top of red-black trees (Cormen et al., 2009, Chapters 13–14). Using these data structures, the querying of elements and/or combinations of attributes and values, which involves operations on intervals, has linear time complexity. Moreover, ancestor-descendant relationships can be expressed in queries and computed in linear time since interval trees are indices supporting efficient structural join operations (Chien et al., 2002). All XML queries other than structural joins can be externally dealt with XPath or XQuery processors. Textual and structural indexes can interact being it possible to query positions within particular regions, regions containing particular positions and combinations of both, all of them computed in linear time.

Besides the design decisions on the algorithms and the data structures used by the backend, there are some system engineering techniques with a direct impact on the backend performance, as the mapping of data structures on memory in order to reduce I/O operations on disk, or the use of RAID technologies to store the indices. What is more, more than ten years of use and development have accumulated a valuable set of improvements balancing size, speed and query complexity. For instance: there is no query optimization implemented but the query evaluation mechanism has been made multithreaded. Another example is regular expressions, which are allowed in queries and are matched against the keys resulting from the sequential exploration of indices. This process can be quite slow when the number of values of the index is very high, as it is for the word-form and lemma indices. In those cases, the values of an attribute are compactly represented as an in-memory minimal acyclic automaton allowing matching to be implemented by enumerating the strings produced by the intersection of the regular expression with the automaton of the corresponding index.

3. Scaling up the corpus backend with MapReduce

For many decades, the doubling of the density of circuits in every computer generation, known as the Moore's Law, has contributed to increases in speed of single-processor machines. Processors speed have benefited from faster clocks, deeper pipelines, and superscalar architectures. However, over the last few years, the lack of opportunities for improving the processors' performance and the possibility to maintain lower power while doubling the speed of many applications has led the industry to a shift to multicore microprocessor designs as the principal strategy for continuing performance growth. Unfortunately, on the storage side, despite disk space having grown at a good pace, latency and bandwidth have improved relatively little.

When data size or processing needs are beyond the capability of a single machine, a cluster of computers is usually required. However, according to Appuswamy et al. (2013), the conventional wisdom in industry and academia for data analytics is that scaling out using a cluster of commodity machines is better than scaling up by adding more resources to a single server. Popular infrastructures such as Hadoop, which were originally designed for petabyte batch processing, are being used in real-world analytics to process jobs with less than 100 Gb of data, with an average size of 14 Gb. They have proven that a single scaled up server with Hadoop can process those jobs and do it as well or better than a with a cluster in terms of performance, cost, power and server density.

MapReduce is often defined as a parallel programming model inspired by functional programming, which has been found useful and easy to apply to practical problems. It is also a design pattern that hides many of the system-level details from the developer. To date, MapReduce is the most successful abstraction applied to large-scale datasets. Scaling up with shared-memory MapReduce eliminates the bottlenecks of scaling out with cluster-based MapReduce, eliminating disk I/O and network latency. However, shared-memory MapReduce is still influenced by other workloads as the key-value memory layout, memory allocation and the framework overhead.

Phoenix++ is a C++ MapReduce implementation for shared-memory multithreaded systems. It has been designed to allow users to write high-performance code easily, with scalability comparable to hand-coded threads solutions (Talbot et al., 2011). Phoenix++ has a modular pipeline providing flexible intermediate key-value storage abstractions allowing workload-tailored implementations and a more efficient combiner implementation minimizing memory usage. The following section presents some results from the implementation of some corpus basic operations in the indexer and the search engine using Phoenix++.

4. Experiments with MapReduce

In order to carry out some experiments with very large corpora and provide a benchmark for other systems, we have used the entire online archive of the Spanish newspaper *El País*[8], covering the period 1976–2011. After performing some data cleansing and part-of-speech tagging, the size of the corpus reaches almost 878 million tokens, including punctuation, in 2.3 million news documents. Additionaly, nearly 23 million structural elements are present as XML/HTML markup in the news. In addition to publi-

[8] http://elpais.com/diario

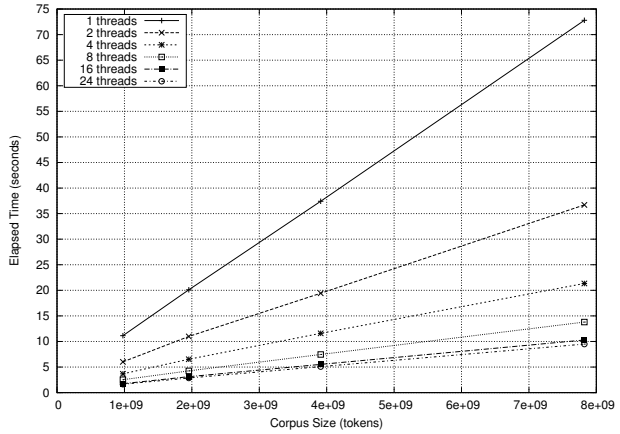


Figure 1: Elapsed time (seconds) for computing word counts as a function of corpus size (tokens).

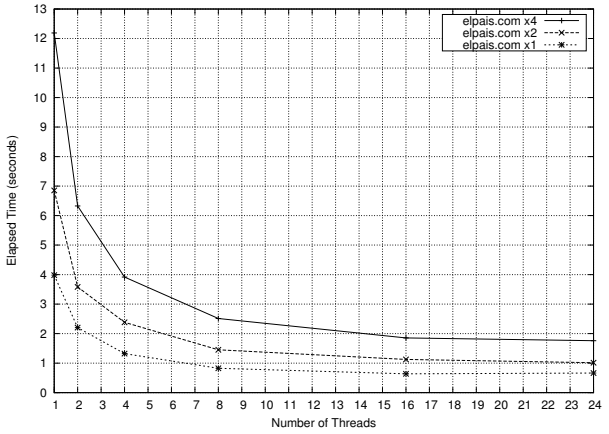


Figure 2: Elapsed time (seconds) for computing word counts as the number of threads is increased.

cation date and newspaper section, each piece of news includes some valuable metadata in the form of descriptors or index terms. These keywords are the data which enable social, political or cultural analysis in the spirit of Culturomics (Michel et al., 2011).

Several corpora have been artificially constructed by appending two, four, eight and even sixteen copies of *El País*, the biggest one reaching nearly fifteen billion words. Although it can be argued that corpora generated in this way are not representative of natural language, in the sense that corpus and vocabulary sizes do not follow Heap's Law, we are using them to approximate results on naturally-constructed corpus of similar size.

Experiments have been conducted on a Dell PowerEdge R615 server with 256 Gb of RAM, two Xeon E5-2620 2 GHz CPUs (24 threads), and 7.200 rpm SAS disks in RAID Level 5 configuration, running Scientific Linux 6.5.

4.1. Word counts

The most basic operation on corpora is word counting. Full word profiles using a different number of threads in MapReduce have been computed ten times for each corpus. The relation between averaged elapsed time, corpus size and number of threads employed is given in Figs. 1 and 2. The first thing to note in Fig. 1 is that, irrespectively of the number of threads used, time scales linearly to the size of corpus, which is an expected and desirable result. Note that complete word profiles for four and eight billion word corpus are computed in five and ten seconds respectively.

Another observation that can be made is about the number of threads used in the computation of profiles and its performance. As Fig. 1 shows, the slope of the straight lines becomes gradually more horizontal as the number of threads is increased, which is equivalent to the slow descent of time observed on the right side of the plot of Fig. 2. We have to consider that the server is running other processes and that as we approach the number of physical threads, the performance of each thread decreases.

4.2. Word cooccurrences

Word cooccurrences is a starting point for a large body of work in lexical distributional semantics including collocational analysis, word sense induction or semantic word vectors. In order to measure the contribution of MapReduce in reducing computing times, the cooccurrences of the high frequency preposition '*de*' (English 'of') have been computed for distances one, three, five and ten.

Fig. 3 shows the variation of elapsed time when the distance, expressed as a radius, grows. Running time has been averaged over ten runs and computed using a different number of threads on the basis of the *Elpais.com* corpus. Note that the word '*de*' appears more than 55.7 million times in *Elpais.com* and that computing the frequency of all its cooccurrences at a maximum distance of ten, on both left and right sides, takes less than two seconds using twenty-four threads.

How word frequency affects running time is shown in Fig. 4, where times for frequencies of '*de*' in multiples (one, two, four and eight) of the base corpus have been computed. As can be seen in the figure, the growth in running time as the frequency is increased is not linear at all, but computing cooccurrences at distance ten for a word appearing 450 million times takes less than eight seconds.

As for word counting, the performance of threads computing cooccurrences decreases as the number of threads used by MapReduce approximates the number of physical threads.

4.3. N-gram frequencies

Statistics on n-grams is another important building block in many linguistic applications. Google and Microsoft have made some statistics on fragments of the Web for n-grams up to length five available. However, idioms, common expressions and named entities need longer n-grams to be captured.

To compute statistics on variable length n-grams we have adapted to Phoenix++ the implementation of the SUFFIX-σ algorithm, which was originally designed for Hadoop (Berberich and Bedathur, 2013). The algorithm applies MapReduce's grouping and sorting functionality only to longest n-grams. Shorter n-grams and their frequencies are obtained from the prefixes of the ordered maximal n-grams. Using Hadoop over a cluster of ten server-class computers with the one billion word New York Times Annotated Cor-

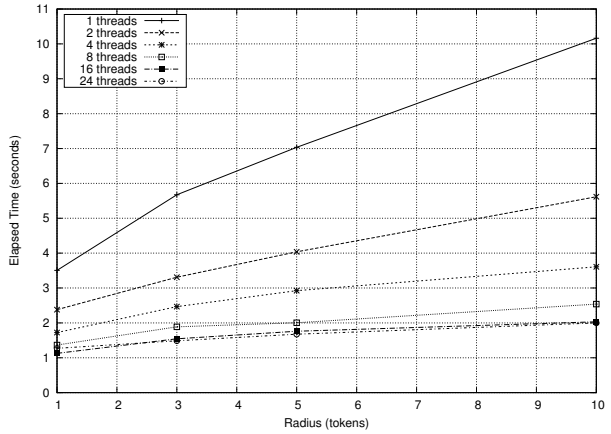


Figure 3: Elapsed time (seconds) for computing the coocurrences of '*de*' in *Elpais.com* as the radius is increased (tokens).

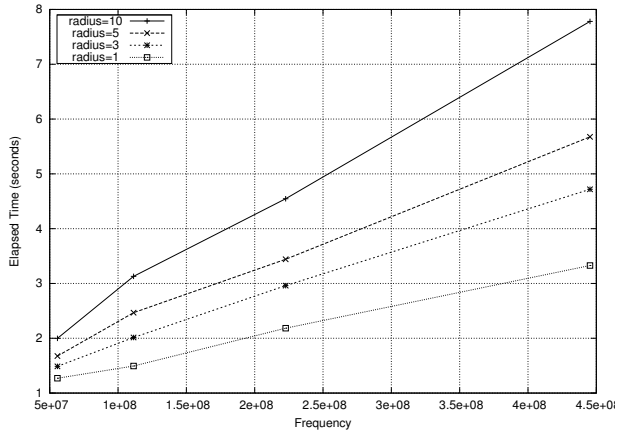


Figure 4: Elapsed time (seconds) using twenty-four threads for computing the coocurrences of '*de*' as a function of its frequency in different size corpora (consisting in several copies of *Elpais.com*).

pus[9], arbitrary length n-grams with a minimum frequency of five were obtained by this algorithm in less than six minutes.

Cluster-based MapReduce solutions to n-grams such as Brants et al. (2007) or Berberich and Bedathur (2013) use hashing on the first or the two first words to bin n-grams into partitions more evenly. In the case of Phoenix++, it provides hash tables for the workload of n-grams and we have hashed whole sequence.

In order to measure the performance of the SUFFIX-σ implementation with our shared-memory version of MapReduce we have used *Elpais.com*'s news metadata to apply the algorithm to samples of different size. As can be seen in Fig. 5, results do not always scale linearly and depends on the particular sample. The performance is worse than that obtained by Berberich and Bedathur (2013) scaling out the problem. This result suggests that there could be still room for improvement, perhaps by implementing other containers or using more adequate hashing functions.

[9] http://catalog.ldc.upenn.edu/LDC2008T19

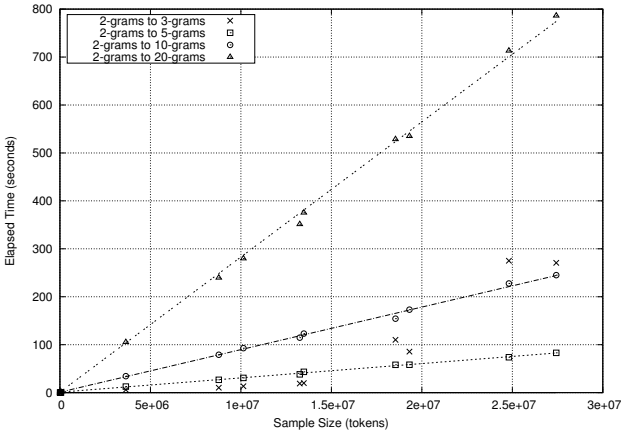


Figure 5: Elapsed time (seconds), using twenty-four threads and the SUFFIX-σ algorithm, for computing n-grams as a function of sample size (tokens).

4.4. File inversion

File inversion is one of the first and most paradigmatic examples of use of cluster-based MapReduce (Lin and Dyer, 2010). To evaluate the implementation of file inversion using shared-memory MapReduce we have indexed several multiples of *Elpais.com* corpus up to thirty-two.

On the one hand, the indexing of text poses no problems to its implementation with MapReduce since the aggregation of posting lists can be implemented as simply concatenation of those lists. On the other hand, the indexing of the structure can also be implemented easily with MapReduce since interval trees emitted during the map phases can be combined and reduced with the join operation on red-black trees, which have a running time proportional to the difference in tree height (Cormen et al., 2009, Exercise 13-2).

A corpus containing sixteen consecutive copies of *Elpais.com*, totalling 15.6 billion words and 36.5 million documents, has been indexed using MapReduce. In order to avoid exhausting memory, indices are dumped to disk every million documents, corresponding roughly to 400-500 million words. Running time for text indexing, annotated with lemmas and morphological analyses from part-of-speech tagging, but without considering the structure, is shown in Fig. 6. As is apparent, time scales well but not linearly with the size of corpora. However, it has been observed that disk access becomes a bottleneck when many threads are simultaneously reading files on the same disk.

5. Conclusions

According to Lin and Dyer (2010), the more observations we gather about language use, the more accurate a description we have of the language itself. However, practical applications must be able to scale up to the size of the datasets of interest. Scaling up with shared-memory MapReduce on multicore computers clearly reduces processing time and scales well with corpus size, making it possible to index corpora with size in the order of tens of billions of words and to obtain basic statistics of interest on corpora beyond the few billion words.

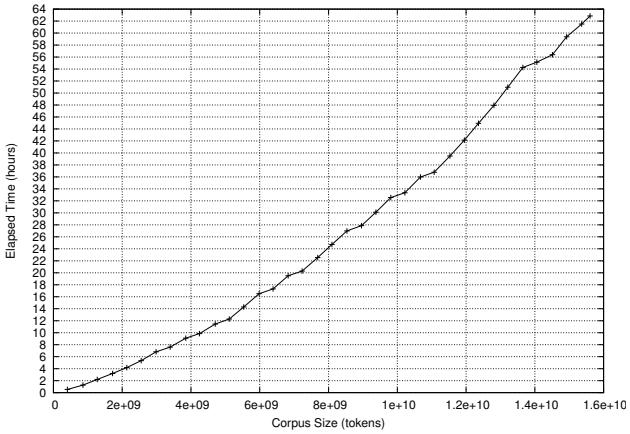


Figure 6: Elapsed time (hours) for text indexing as a function of corpus size (tokens). Samples taken at intervals of 400-500 million tokens, after saving indices to disk.

6. Acknowledgements

We want to thank our colleagues of the Computational Linguistics area of the RAE for the millions of queries over almost fifteen years of the backend's lifespan, for using it sometimes in strange ways, and for considering the use of the backend in almost every project we started, even when it wasn't a priori the best option.

7. References

Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., and Rowstron, A. (2013). Scale-up vs scale-out for Hadoop: Time to rethink? In *Proceedings of the 4th Annual Symposium on Cloud Computing*, pages 20:1–20:13, New York, NY, USA. ACM.

Berberich, K. and Bedathur, S. (2013). Computing n-Gram Statistics in MapReduce. In *16th International Conference on Extending Database Technology, EDBT '13*, Genua, Italy.

Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867.

Chien, S.-Y., Vagena, Z., Zhang, D., Tsotras, V. J., and Zaniolo, C. (2002). Efficient structural joins on indexed XML documents. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, pages 263–274.

Christ, O. (1994). A modular and flexible architecture for an integrated corpus query system. In *Proceedings of the COMPLEX '94*, pages 23–32.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. The MIT Press, 3rd edition.

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.

Davies, M. (2009). The 385+ Million Word Corpus of Contemporary American English (1990–present). *International Journal of Corpus Linguistics*, 14(2):159–190.

Davies, M. (2010). More than a peephole: Using large and diverse online corpora. *International Journal of Corpus Linguistics*, 14(2):159–190.

ISO:24612. (2012). Language resource management – Linguistic annotation framework (LAF). Technical Report ISO 24612, ISO.

Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlý, P., and Suchomel, V. (2013). The TenTen Corpus Family. In *7th International Corpus Linguistics Conference*, Lancaster.

Janus, D. and Przepiórkowski, A. (2007). Poliqarp: An open source corpus indexer and search engine with syntactic extensions. In *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*.

Kupietz, M., Belica, C., Keibel, H., and Witt, A. (2010). The German reference corpus DeReKo: A primordial sample for linguistic research. In *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC-2010)*, pages 1848–1854, Valletta, Malta.

Lin, J. and Dyer, C. (2010). *Data-Intensive Text Processing with MapReduce*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., The Google Books Team, Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. (2011). Quantitative analysis of culture using millions of digitized books. *Science*, 331.

Porta, J. and Ruiz-Ureña, R. J. (2003). Lexicometría de corpus. *Procesamiento del Lenguaje Natural*, 31.

Rychlý, P. (2000). *Korpusové manažery a jejich efektivní implementace (Corpus Managers and their effective implementation)*. Ph.D. thesis, Fakulta Informatiky. Masarykova Univerzita, Brno.

Rychlý, P. (2007). Manatee/Bonito - A Modular Corpus Manager. In *First Workshop on Recent Advances in Slavonic Natural Language Processing*, pages 65–70, Brno.

Schneider, R. (2012). Evaluating RDBMS-based access strategies to very large multi-layer corpora. In *Proceedings of the LREC 2012 Workshop: Challenges in the management of large corpora*, Istanbul, Turkey.

Schnober, C. (2012). Using information retrieval technology for a corpus analysis platform. In *Proceedings of KONVENS '12*, pages 199–207, September.

Talbot, J., Yoo, R. M., and Kozyrakis, C. (2011). Phoenix++: Modular MapReduce for shared-memory systems. In *Proceedings of the Second International Workshop on MapReduce and its Applications*, pages 9–16.

Zobel, J. and Moffat, A. (2006). Inverted files for text search engines. *ACM Computer Surveys*, 38(2).

Text Corpora for Text Studies. About the foundations of the AAC-Austrian Academy Corpus.

Hanno Biber
Institute for Corpus Linguistics and Text Technology / Literature in Transition.
Austrian Academy of Sciences
hanno.biber@oeaw.ac.at

Evelyn Breiteneder
Institute for Corpus Linguistics and Text Technology / Literature in Transition.
Austrian Academy of Sciences
evelyn.breiteneder@oeaw.ac.at

Abstract

One of the primary research aims of the research group at the "LIT – Literature in Transition" research group at the "Austrian Academy of Sciences" is to develop large text language resources for the study of literary texts. The paper will give an answer to the question how a significant historical text corpus of a considerable size that has been built along the principles of corpus research can be made use of for practical research purposes in the field of literary text studies. The potential of a corpus-based methodological approach of literary text studies will be presented by investigating the textual qualities and the specific language use of literary texts. The paper will present several aspects of the research question illustrated by text examples and discuss the methodological implications of such a corpus based investigation thereby facing particular challenges with regard to large text corpora. Literary texts have properties that can be recognized and registered by means of a corpus-based study of language. Therefore the literary texts of the "AAC-Austrian Academy Corpus", a diachronic German language digital text corpus of more than 500 million tokens, will be used as examples for this research, thus constituting an ideal text basis for a corpus linguistic exploration into the fields of lexicographic units, syntactic constructions, narrative procedures and related issues concerned with the study of literary texts.

Keywords: Corpus Linguistics, Literature, Text Studies

1. Text Corpora

The "AAC - Austrian Academy Corpus" is a digital diachronic German language text corpus of more than 500 million tokens. Within this corpus texts from the literary domain constitute a significant portion of the overall holdings. The text corpus comprises several thousands of texts representing a wide range of different text types. Therefor the question of structuring these large amounts of texts for the purpose of literary studies and literary analysis is quite demanding. One of the primary research aims of the research group at the "LIT – Literature in Transition" research group at the "Institute for Corpus Linguistics and Text Technology" at the "Austrian Academy of Sciences" in Vienna is to develop large text language resources for the study of literary texts. This paper will give an overview of the latest initiatives in the relatively new field of "LIT – Literature in Transition", whereby the latest developments in the fields of both digital and digitalized literature will be taken into consideration. In addition to that, based upon the ICLTT's researchers' expertise in dealing with large historical text corpora like the "AAC- Austrian Academy Corpus" answers to the question will be given, how a significant historical text corpus of a considerable size that has been built along the principles of corpus research can be made use of for practical research purposes in the field of literary text studies. The potential of a corpus-based methodological approach of literary text studies will be presented by investigating the textual qualities and the specific language use in the texts of the corpus, which have been annotated and made accessible with the help of corpus linguistics. The studies discussed will focus on

several aspects of the research questions illustrated by text examples and discuss the methodological implications of a corpus-based investigation into the use of language in literary texts. Literary texts have properties that can be recognized and registered by means of a corpus-based study of language. Therefore the literary texts of the "AAC-Austrian Academy Corpus" will be used as examples for this research. This source is an ideal text basis for a corpus linguistic exploration into the fields of specific lexicographic units, syntactic constructions, narrative procedures and related issues concerned with the study of literary texts. The corpus-based approach allows various ways of philological research and text analysis.

For corpus linguistics and corpus based language research large text corpora need to be structured in a systematic way. For this structural purpose the AAC is making use of the notion of container. By container in the context of corpus research we understand a flexible system of pragmatic representation, manipulation, modification and structured storage of annotated items of text. The issue of representing a large corpus in formats that offer only limited space is paradigmatic for the general task of representing a language by just a small collection of text or a small sample of the language. Methods based upon structural normalization and standardization have to be developed in order to provide useful instruments for text studies. This general aspect is all the more valid for a large subcoprpus of literary texts with manifold texttypes and genres that have to be accounted for.

The "AAC-Austrian Academy Corpus" texts integrated into the AAC stemming from the last 150 years are predominantly German language texts and are of considerable historical and cultural significance. Among the AAC's sources, which cover manifold domains and genres, there are literary journals, newspapers, novels, dramas, poems, advertisements, essays, travel accounts, cookbooks, pamphlets, political speeches as well as plenty of scientific, legal, religious texts, to name just a few. The historical period covered by the corpus is ranging from the 1848 revolution to the fall of the iron curtain in 1989. In this period significant historical changes with remarkable influences on the language and the language use can be observed. Thousands of language documents and literary objects by thousands of authors have been collected, representing an astonishing range of different text types from all over the German speaking areas. The AAC corpus holdings provide a great number of reliable resources and interesting corpus based approaches for investigations into the linguistic and textual properties of these texts. More than 500 million running words of text have been scanned, converted into machine-readable text and annotated with basic structural mark-up and selected thematic mark-up.

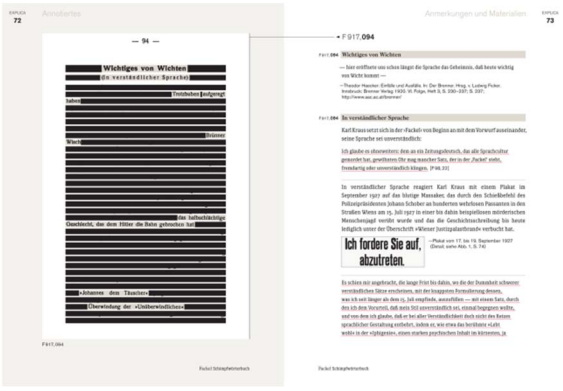
The research will have to focus on methods and resources for making large amounts of texts accessible in a well-structured way. The AAC Container, a systematic central and well-structured access point to the holdings of the entire corpus, will make more detailed investigations into the specific qualities of the texts possible.

In contrast to other corpus-oriented projects, the AAC proceeds from literary and text lexicographic research. Corpus research and the creation of large electronic text collections have traditionally been the domain of linguists. Literary digitization initiatives were often restricted to particular writers, many of these projects did neither produce large amounts of data nor pursue research on methods of how to tackle the problems involved in working with such data. Being aware of the need of digital resources in many fields of the humanities, the AAC has started to work on applications, tools and methods geared towards a wider range of applications trying to pursue a path of text-oriented computing. While the needs of linguists have not been ignored, they tried to work towards applications that also offered access to coherent texts, convinced that for many applications it is indispensable for researchers to have access to the text as such.

The AAC tools support this technology which allows both the controlled application of markup as well as automated validation of large amounts of data. The AAC applies encoding schemes characterized by a combined approach to capture both structural features of the texts as well as a certain amount of data describing the physical appearance of the original texts. This approach has led to hybrid systems of markup not only representing the basic semantic structures of the texts but also layout information and information about the graphic design and the typographical semantics. In digitizing historical data, semantic and presentational data will remain intertwined. When working on large amounts of such texts, it is considerably easier to capture formal data than to translate typographic idiosyncrasies into consistent structural markup. The AAC collects descriptive metadata concerning the digitized objects that is stored in a relational database containing around 7000 records holding all relevant information about the physical objects. Having established a working infrastructure for the digital texts available, the AAC is developing more sophisticated methods of utilizing large scale corpora on the basis of various database systems as well as XML-aware indexing tools to establish standard procedures of accessing large XML text repositories. The AAC's digitizing activities are characterized by a strong connection to the physical objects of digitization. This may be seen as one of the motives behind the one-page-one-file principle, which implies that each page of a printed publication is stored as a separate digital object in the digital medium which - by means of implicit and explicit linking - form larger digital objects that in turn represent coherent texts. By doing so, the AAC attaches importance to the semantic structures of the text as well as to the physical appearance of the text.

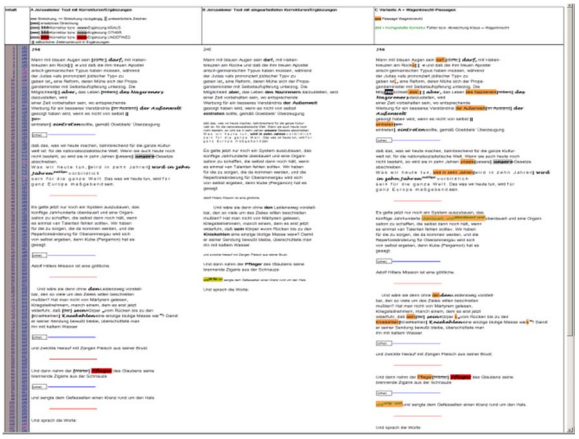
2. Text Studies

The satire of Karl Kraus and his use of phraseological language is of interest here with regard to the use of idioms and multiword units and in particular the satirical treatment of the language phenomena. The study of phraseological and pejorative language use is also related to the more general methodological questions of language and phraseology as research topics for corpus research. Its findings of significant examples will be complemented with the lexicographic evidence and the documentation of the use of expressions to be found in the Dictionary of Idioms, "Wörterbuch der Redensarten zu der von Karl Kraus 1899 bis 1936 herausgegebenen Zeitschrift 'Die Fackel'" published in 1999.



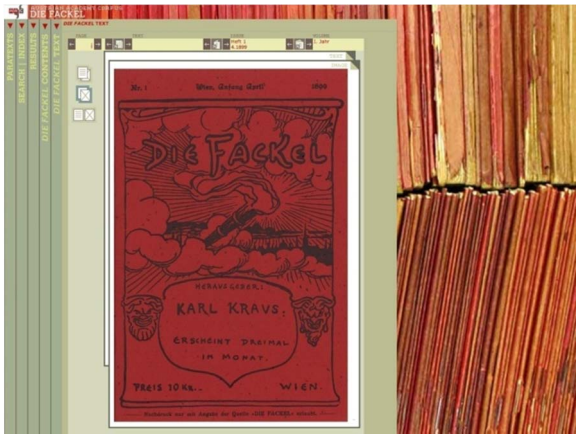
In 2008 the text-dictionary of pejorative expressions, the Dictionary of Invectives, "Schimpfwörterbuch zu der von Karl Kraus 1899 bis 1936 herausgegebenen Zeitschrift 'Die Fackel'" followed. Both dictionaries were built

upon corpus methods and offer specific insights into the research questions mentioned above concerning idioms, proverbs, lexicalized metaphors, figurative compounds and the like.



In a third book project based upon the literary texts by Karl Kraus emphasis will be given to the "Dritte Walpurgisnacht" (Third Walpurgis Night) written by the satirist in 1933, which will be taken in digital format, among other sources, as a starting point for text analysis and as concerns the AAC also as a starting point for text selection for the corpus. This text has to be regarded as the most important contemporary text of German literature dealing with National Socialism. In the "Dritte Walpurgisnacht" Karl Kraus has documented the murderous reality of the Nazi regime as early as May 1933 and documented and commented upon the murderous language of that time in numerous examples.

Two AAC projects have to be mentioned in this paper on issues of providing research instruments based upon principles of corpus linguistics, the digital editions of the literary journals "Die Fackel" and "Der Brenner", the AAC-FACKEL and BRENNER ONLINE. How can a valuable historical text source based upon the principles of corpus research be used for research in the field of literary studies? One important feature to achieve this goal is to provide an interface that will function as a reading instrument and as a search instrument at the same time. The edition interface for the AAC-FACKEL, like the second one, Brenner online, has five individual frames synchronized within one single window. The frames can be opened and closed as required. The paratext section situated within the first frame provides additional information about the background of the edition and scholarly essays about the journal. The contents section provides access to the whole run of issues and all of the contents of the journal in chronological order. The text section has a complex and powerful navigational bar at the top so that the reader can easily navigate and read within the journal either in text-mode or in image-mode from page to page, from text to text, from issue to issue, and with the help of hyperlinks. Creating an intuitive navigable structure in retrodigitized historical journals poses a number of problems on top of which is the compilation of contents lists. The readers can navigate the texts in four ways in this digital edition: from page to page, from text to text, from issue to issue, from volume to volume. The readers can read the text in two ways: page by page as digital text or as facsimile image. The search and index section gives access to a variety of indexes, databases and full-text search mechanisms. The results of these queries and lists are displayed in the adjacent results section. These digital editions will function as models for similar applications for the access to corpus based text studies provided for scholars and the interested public alike. They provide well-structured and well-designed access to the sources. After a thorough survey of all periodicals mentioned in the journal, this data was disambiguated and enriched with external data. And linguistic data produced were restricted to lemmatization and the assignment of word class information. POS and lemma data was created by a standard tagger.



To do literary and historical research it is indispensable to have access to the texts in their entirety. Therefore, access is offered to the corpus not only through query result lists but also through a library unit which allows readers to navigate to any desired part of the corpus. The AAC corpus browser tools have to be seen as work for the AAC Container, by which the working group understands the framework for the future presentation of the corpus in particular for the literary journals section of the entire corpus.

3. References

AAC-Austrian Academy Corpus: AAC-FACKEL. Online Version: »Die Fackel. Herausgeber: Karl Kraus, Wien 1899-1936«. AAC Digital Edition No 1 (Editors-in-chief: Hanno Biber, Evelyn Breiteneder, Heinrich Kabas, Karlheinz Mörth), http://www.aac.ac.at/fackel

AAC-Austrian Academy Corpus and Brenner-Archiv: BRENNER ONLINE. Online Version: »Der Brenner. Herausgeber: Ludwig Ficker, Innsbruck 1910-1954«, AAC Digital Edition No 2, (Editors-in-chief: Hanno Biber, Evelyn Breiteneder, Heinrich Kabas, Karlheinz Mörth), http://www.aac.ac.at/brenner

Biber, Hanno; Breiteneder, Evelyn (2012): Fivehundredmillionandone Tokens. Loading the AAC Container with Text Resources for Text Studies. In: Proceedings of LREC 2012. 8th Language Resources and Evaluation Conference, Main Conference, Istanbul 23.-25. 5. 2012. Ed. By Calzolari, Nicoletta; Choukri, Khalid; Declerck, Thierry; Mariani, Joseph. Istanbul LREC 2012, p. 38 (print and online).

Biber, Hanno; Breiteneder, Evelyn (2012): The AAC Container: Managing Text Resources for Text Studies. In: Proceedings of LREC 2012. 8th Language Resources and Evaluation Conference, Workshop, Istanbul 23.-25. 5. 2012. Ed. Banski, Piotr; Kupietz, Marc; Witt, Andreas et al. Istanbul LREC 2012, p. 43-44 (print and online).

Welzig, Werner (2008): Schimpfwörterbuch zu der von Karl Kraus 1899 bis 1936 herausgegebenen Zeitschrift 'Die Fackel' [in 3 volumes:] Alphabetisches, Chronologisches, Explikatives. Austrian Academy Press, Vienna.

Welzig, Werner (1999): Wörterbuch der Redensarten zu der von Karl Kraus 1899 bis 1936 herausgegebenen Zeitschrift 'Die Fackel'. Austrian Academy Press, Vienna.