Hamburg: Universität Hamburg, 2011. pp. 87-92. (Working Papers In Multilingualism Series B 96)

A functional database framework for querying very large multi-layer corpora

Roman Schneider
Institut für deutsche Sprache
R5 6-13, D-68161 Mannheim
schneider@ids-mannheim.de

Abstract

Linguistic query systems are special purpose IR applications. We present a novel state-of-the-art approach for the efficient exploitation of very large linguistic corpora, combining the advantages of relational database management systems (RDBMS) with the functional MapReduce programming model. Our implementation uses the German DEREKO reference corpus with multi-layer linguistic annotations and several types of text-specific metadata, but the proposed strategy is language-independent and adaptable to large-scale multilingual corpora.

Keywords: corpus storage, multi-layer corpora, corpus retrieval, database systems

1. Introduction

In recent years, the quantitative examination of natural language phenomena has become one of the predominant paradigms within (computational) linguistics. Both fundamental research on the basic principles of human language, as well as the development of speech and language technology, increasingly rely on the empirical verification of assumptions, rules, and theories. More data are better data (Church, & Mercer, 1993): Consequently, we notice a growing number of national initiatives related to the building of large representative datasets for contemporary world languages. Besides written (and sometimes spoken) language samples, these corpora usually contain vast collections of morphosyntactic, phonetic, semantic etc. annotations, plus text- or corpus-specific metadata. The downside of this trend is obvious: Even with specialized applications, our ability to store linguistic data is often bigger than the ability to process all this data.

A lot of essential work towards the querying of linguistic corpora goes into data representation, integration of different annotation systems, and the formulation of query languages (e.g., Rehm et al., 2008; Zeldes et al., 2009; Kepser, Mönnich & Morawietz, 2010). But the scaling problem still remains: As we go beyond corpus sizes of some million words, and at the same time increase the number of annotation systems and search keys, query costs rise disproportionately. This is due to the fact that unlike traditional IR systems, corpus retrieval systems not only have to deal with the "horizontal" representation of textual data, but with heterogeneous metadata on all levels of linguistic description. And, of course, the exploration of inter-relationships between annotations becomes more and more challenging as the number of annotation systems increases. Given this context, we present a novel approach to scale up to billion-word corpora, using the example of the multi-layer annotated German Reference Corpus DEREKO.

2. The Data

The German Reference Corpus DEREKO currently comprises more than four billion words and constitutes the largest linguistically motivated collection of contemporary German. It contains fictional, scientific, and newspaper texts – as well as several other text types – and is annotated morphosyntactically with three competing systems (Connexor, Xerox, TreeTagger). The automated enrichment with additional metadata is underway.

	1 Mio	10 Mio	100 Mio	1000 Mio	4000 Mio
rare (*traurige/isoliert/trauen*)	0,03s	0,16s	0,3s	0,6s	0,9s
low (*langsam/lesen/verfügt*)	0,29s	0,35s	0,37s	0,65s	1s
mid (*ohne/nun/uns*)	0,3s	0,4s	0,5s	1,2s	3,8s
high (*nicht/ist/dem*)	0,31s	0,47s	1,49s	10,47s	38,7s
top (*der/die/und*)	0,4s	0,87s	4,67s	47,06s	301s

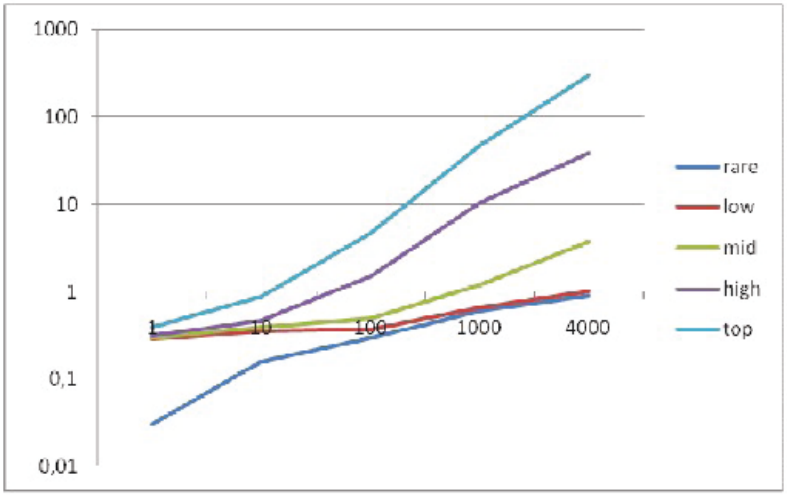

Figure 1: Response times for nested SQL queries with three search keys (logarithmic scaled axis)

3. Existing Approaches

We empirically evaluated the most prominent existing querying approaches, and contrasted them with our functional model (the full paper will contain our detailed series of measurements). Given the reasonable assumptions that XML/SGML-based markup languages are more suitable for data exchange than for efficient storing and retrieval, and that traditional file-based data storage is less robust and powerful than database management systems, we focused on the following strategies:

i. In-Memory Search: Due to the fact that a computer's main memory is still the fastest form of data storage, there are attempts to implement in-memory databases even for considerably large corpora (Pomikálek, Rychlý & Kilgarriff, 2009). These indexless systems perform well for unparsed texts, but are strongly limited in terms of storage size and therefore cannot deal with data-intensive multi-layer annotations.

ii. N-Gram Tables: In order to overcome physical limitations, newer approaches use database management systems and decompose sequences of strings into indexed n-gram tables (Davies, 2005). This allows queries over a limited number of search expressions, but space requirements for increasing values of n are enormous. Sentence-external queries with regular expressions or NOT-queries – both are crucial for comprehensive linguistic exploration – cannot use the n-gram-based indexes and thus perform rather poor.

iii. Advanced SQL: Another strategy is to make use of the relational power of sub-queries and joins within a RDBMS. Chiarcos et al. (2008) use an intermediate language between query formulation and database backend; Bird et al. (2005) present an algorithm for the direct translation of linguistic queries into SQL. This approach uses absolute word positions, and therefore allows proximity queries without limitation of word distances. But again, even with the aid of the integrated cost-based optimizer (CBO), response times for increasing numbers of search keys become extremely long. We evaluated the proposed strategy on 1, 10, 100, 1000,

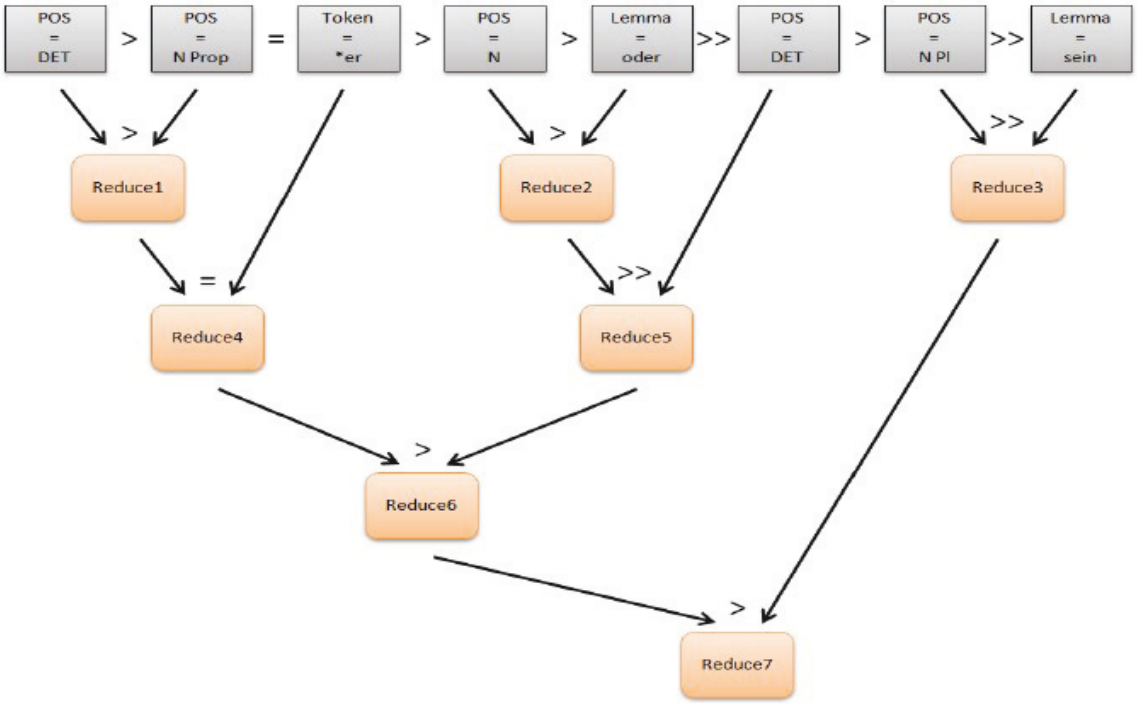

Figure 2: MapReduce processes for a concatenated query with eight search keys

iv. and 4000 million word corpora with rare-, low-, mid-, high-, and top-level search keys and found out that concatenated queries soon exceed the capability of our reference server because nested loops generate an immense workload. Figure 1 shows the response times in seconds for the query "select count(t1.co_sentenceid) from tb_token t1, (select co_id, co_sentenceid from tb_token where co_token=token1) t3, (select co_id, co_sentenceid from tb_token where co_token = token2) t2 where co_token = token3 and t1.co_sentenceid = t2.co_sentenceid and t1.co_sentenceid = t3.co_sentenceid and t1.co_id > t2.co_id and t2.co_id > t3.co_id;", using three search keys on identical metadata types and a single-column index. This query simply counts the number of sentences that contain three specified tokens (token1, token2, token3) in a fixed order. Compared to a similar query on the 4000 Mio corpus with one search key (5s for a top-level search) or two search keys (56s), the increase of response time is obviously disproportional (301s). It gets remarkably less performant for searches on different metadata types (token, lemma, part-of-speech etc.) using multi-column indexes. Furthermore, by adding text-specific metada restrictions like text type or publication year, this querying strategy produces response times of several hours and thereby becomes fully unacceptable for real-time applications.

4. Design and Implementation

As our evaluation shows, existing approaches do not handle queries with complex metadata on very large datasets sufficiently. In order to overcome bottlenecks, we propose a strategy that allows the distribution of data and processor-intensive computation over several processor cores – or even cluster of machines – and facilitates the partition of complex queries at runtime into independent single queries that can be executed in parallel. It is based on two presuppositions:

i. Mature relational DBMS can be used effectively to maintain parsed texts and linguistic metadata. We intensively evaluated different types of tables (heap tables, partitioned tables, index organized tables) as well as different index types (B-tree, bitmap, concatenated, functional) for the distributed storing and retrieval of linguistic data.

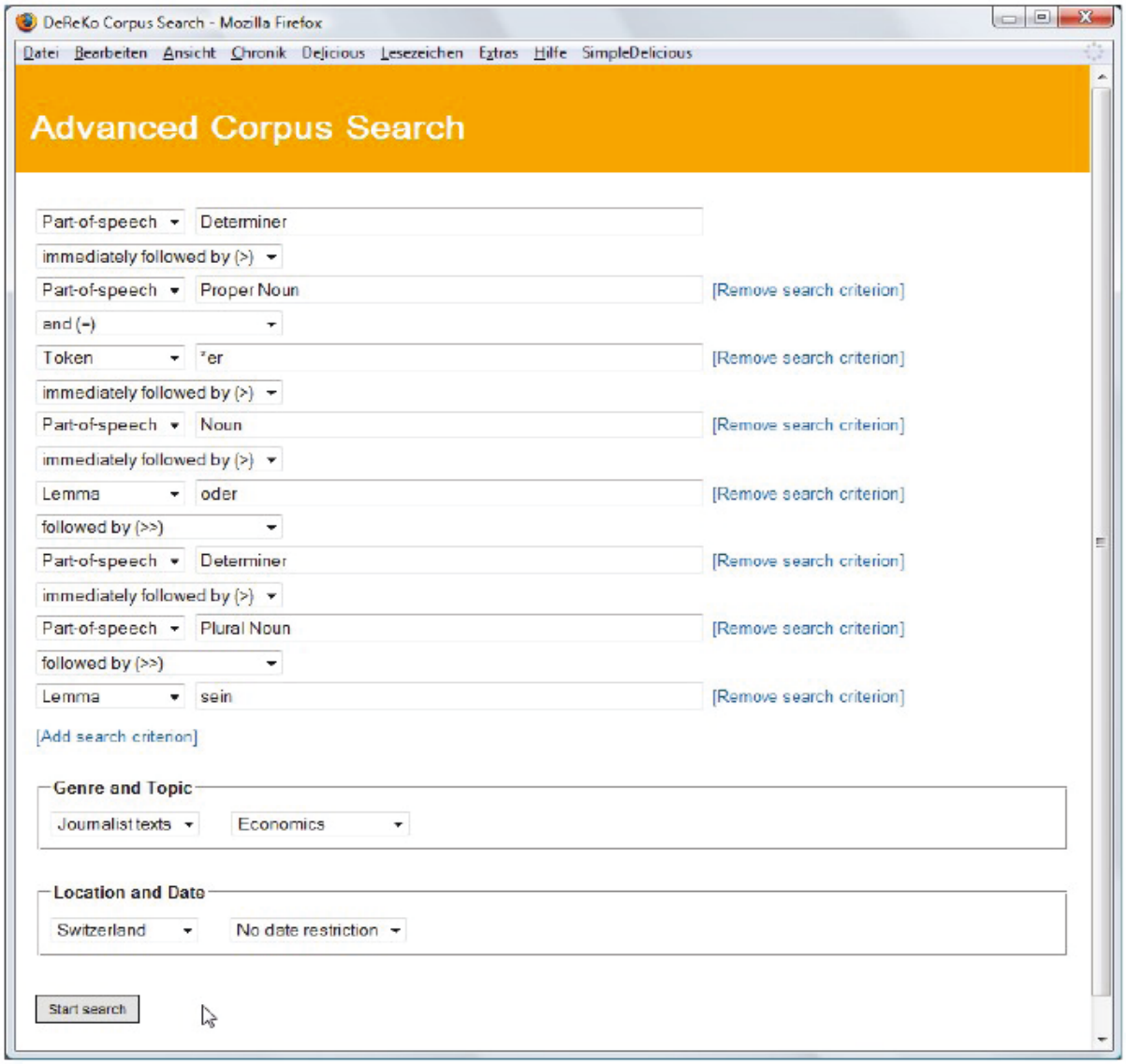

Figure 3: Web-based retrieval form with our sample query

ii. The MapReduce programming model supports distributed programming and tackles large-data problems. Though MapReduce is already in use in a wide range of data-intensive applications (Lin & Dyer, 2010), its principle of "divide and conquer" has not been employed for corpus retrieval yet.

In order to prove the feasibility of our approach, we implemented our corpus storage and retrieval framework on a commodity low-end server (quad-core microprocessor with 2.67 GHz clock rate, 16GB RAM). For the reliable measurement of query execution times, and especially to avoid caching effects, we always used a cold-started 64-bit database engine.)

Figure 2 illustrates the map/reduce processes for a complex query, using eight dictinct search keys on different metadata types: Find all sentences containing a determiner immediately followed by a proper noun ending on "er", immediately followed by a noun, immediately follwed by the lemma "oder", followed by a determiner (any distance), immediately followed by a plural noun, followed by the lemma "sein" (any distance). Within a "map" step, the original query is partitioned into eight separate key-value pairs. Keys represent linguistic units (position, token, lemma, part-of-speech, etc.), values may be the actual content. Thus, we can simulate regular expressions (a feature that is often demanded for advanced corpus retrieval systems, but difficult to implement for very large datasets).

The queries can be processed in parallel and pass their results (sentence/position) to temporary tables. The

subsequent “reduce” processes filter out inappropriate results step by step. Usually, this cannot be executed in parallel, because each reduction produces the basis for the next step. But our framework, implemented with the help of stored procedures within the RDBMS, overcomes this restriction by dividing the process tree into multiple sub-trees. The reduce processes for each sub-tree are scheduled simultaneously, and aggregate their results after they are finished. So the seven reduce steps of our example can be executed within only four parallel stages.

Our concatenated sample query with eight muti-type search keys on a four billion word corpus took less than four minutes, compared with several hours when employing SQL joins as in 3 (iii). The parallel MapReduce framework is invoked by an extensible web-based retrieval form (see figure 3) and stores the search results within the RDBMS, thus making it easy to reuse them for further statistical processing. Additional metadata restrictions (genre, topic, location, date) are translated into separate map processes and reduced/merged in parallel to the main search.

5. Summary

The results of our study demonstrate that the joining of relational DBMS technology with a functional/parallel computing framework like MapReduce combines the best of both worlds for linguistically motivated large-scale corpus retrieval. On our reference server, it clearly outperforms other existing approaches. For the future, we plan some scheduling refinements of our parallel framework, as well as support for additional levels of linguistic description and metadata types.

6. References

Church, K., Mercer, R. (1993): Introduction to the Special Issue on Computational Linguistics Using Large Corpora. Computational Linguistics 19:1, pp. 1-24.

Rehm, G., Schonefeld, O., Witt, A., Chiarcos, C., Lehmberg, T. (2008): A Web-Platform for Preserving, Exploring, Visualising and Querying Linguistic Corpora and other Resources. Procesamiento del Lenguaje Natural 41, pp. 155-162.

Zeldes, A., Ritz, J., Lüdeling, A., Chiarcos, C. (2009): ANNIS: A Search Tool for Multi-Layer Annotated Corpora. Proceedings of Corpus Linguistics 2009. July 20-23, Liverpool, UK.

Kepser, S., Mönnich, U., Morawietz, F. (2010): Regular Query Techniques for XML-Documents. Metzing, D., Witt, A. (Eds): Linguistic modeling of information and Markup Languages, Springer, pp. 249-266.

Pomikálek, J., Rychlý, P., Kilgarriff, A. (2009): Scaling to Billion-plus Word Corpora. Advances in Computational Linguistics 41, pp. 3-13.

Davies, M. (2005): The advantage of using relational databases for large corpora. International Journal of Corpus Linguistics 10 (3), pp. 307-334.

Chiarcos, C., Dipper, S., Götze, M., Leser, U., Lüdeling, A., Ritz, J., Stede, M. (2008): A Flexible Framework for Integrating Annotations from Different Tools and Tag Sets. Traitement Automatique des Langues 49(2), pp. 271-293.

Bird, S., Chen, Y., Davidson, S., Lee, H., Zhen, Y. (2005): Extending Xpath to Support Linguistic Queries. Workshop on Programming Language Technologies for XML (Plan-X).

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