Big Scale Text Analytics and Smart Content Navigation

Karsten Schmidt[1], Sebastian Bächle[1], Philipp Scholl[1], and Georg Nold[2]

[1] SAP AG, Walldorf, Germany,
{karsten.schmidt01|sebastian.baechle|p.scholl}@sap.com, www.sap.com
[2] Springer Science+Business Media, Berlin, Germany, www.springer.com

Abstract. Identifying and exploring relevant content in growing document collections is a challenge for researchers, users, and system providers alike. Supporting this is crucial for companies offering knowledge in the form of documents as their core product. Our demo shows an intelligent way of doing guided research in big text collections, using the collection of the major scientific publisher Springer SBM as an example data set. We use the SAP HANA platform for flexible text analysis, ad-hoc calculations and data linkage, in order to enhance the experience of users navigating and exploring publications. We integrate unstructured data (textual documents) and structured data (document metadata and web server logs), and provide interactive filters in order to enable a responsive user experience while searching for relevant content. With HANA, we are able to implement this functionality over big data on a single machine by making use of HANA's SQL data store and the built-in application server.

Key words: SAP HANA, analytics, information retrieval

1 Context-sensitive Information Retrieval

Helping users to locate relevant information in large text collections requires sophisticated search functionality with meaningful result ranking. The actual information need is often fuzzy and hard to determine in advance. Typically, a research session starts with a simple keyword search, which is incrementally refined and adjusted according to the results delivered. In this demo, we demonstrate an application for exploring a large repository of unstructured content through an advanced integration of full-text search and text analytics. In addition to fast live search with as-you-type results, the application provides user guidance through domain-specific search, context-sensitive linking and content-based recommendations. Beyond the scope of a conventional text retrieval system, our application integrates further data sources, e.g., to improve the quality of search results and recommendations. Furthermore, we include structured web server logs for popularity analytics, a custom vocabulary for searching medical content, and context-specific embedding of third-party content.

Single-Box Architecture Big-scale content retrieval typically requires a multitude of dedicated systems for storage, text processing, and analytics, as well as an application server and a web server. In such a setting, data is stored across multiple machines (see Fig. 1) and costly joins are always unavoidable. Furthermore, different access languages and layers increase latency due to data transformations and translations. Pre-calculated aggregates mitigate these issues, but increase the data volume significantly and cause additional update and maintenance overhead.

Using the HANA platform, all functionality can be consolidated in a single box (see Fig. 2). HANA features a powerful column store, an integrated text analytics engine and a built-in application server, with all data being accessible via SQL. This turns HANA into a convenient, lightweight, and scalable platform for managing structured *and* unstructured content.

Our demo application runs on a HANA server with 80 cores (8 CPUs à 10 cores at 2.5 GHz) and 1 TB main memory. On this machine, response times are between 40 ms for simple search requests and 500 ms for complex recommendations. Due to the tight integration of the built-in application layer, the response times are close to those of plain SQL.

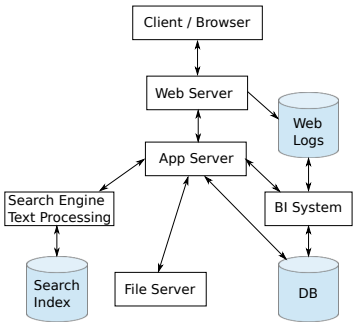

Fig. 1. State-of-the-art architecture

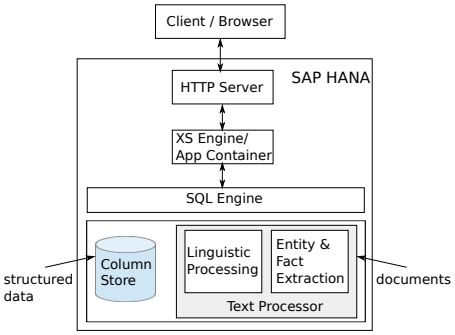

Fig. 2. Single box HANA architecture

The Big Data Challenge Our database consists of 3.7 million scientific articles and book chapters[1], all in PDF format, with a raw data volume of 3.6 TB. Extracting valuable pieces of information from such a large amount of unstructured text data is a challenge in many ways. Aside from standard full-text search, complex text analysis is required to identify relevant terms and topics for similarity search, recommendations, and relevance rankings. Creating, materializing, and maintaining search indexes and pre-computing similarity measures is expensive, and should be avoided for quickly growing document bases. Hence, we avoid pre-computed data structures wherever possible. Our queries operate directly on a few basic data structures and exploit the parallel hardware to compute document statistics and similarities on the fly.

[1] published by Springer Science+Business Media [2]

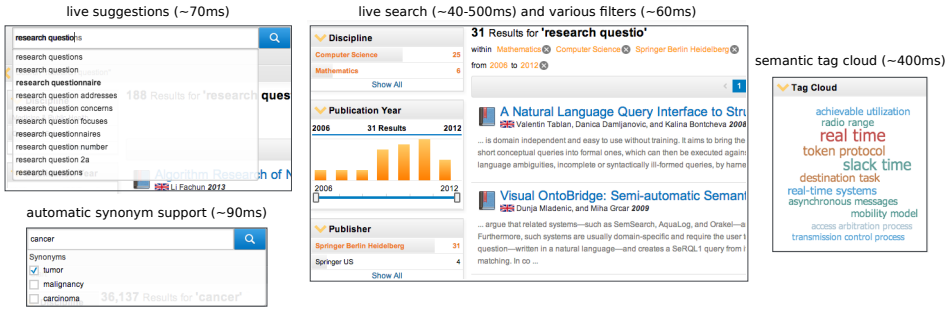

Fig. 3. Search features and response times for demo application.

We create a full-text index and extract more than 2.8 billion text entities from all documents and store them in a plain relational table. As the automatic indexing and entity extraction runs incrementally, new documents can be added without re-indexing and re-extracting.

Computing ad-hoc similarity over the whole collection for giving recommendations is an extremely expensive task and heavily depends on the similarity measure. To solve this issue, a basic momentum table, summarizing entity statistics per document forms our baseline. This small table is cheap to maintain when new documents are added to the collection. Of course, statistics for existing documents are not affected.

Along with the documents, we stored 1 GB of structured metadata with information about authors, publishers, etc. Additionally, we have loaded 14 months (1.3 billion rows) of web server log data into the same database instance. This corresponds to 392 GB of raw log data and 43 GB in the in-memory format.

2 Demo Content

The demo comprises an appealing web application presenting four key aspects:

Search Searching and filtering of *unstructured* text, which is enhanced by *structured* metadata and a domain-specific vocabulary for medical terms (Fig. 3).

Browsing An embedded document viewer provides highlighting of extracted entities, links for inter-document navigation, and ad-hoc analysis and recommendations for freely selectable text fragments (see Fig. 4). The linkage to third-party content is exemplified by integrating the Wikipedia API [3].

Analytics Analytics for *unstructured* data is shown with a live-updating tag cloud and content-based recommendations. Analytics for *structured* data is exemplified by live popularity analysis over the web log data.

Internals The SQL-based backend of the application is exposed using our developer workbench (see Fig. 5). The audience gets direct insight into the base tables, extracted entities, and can issue SQL statements.

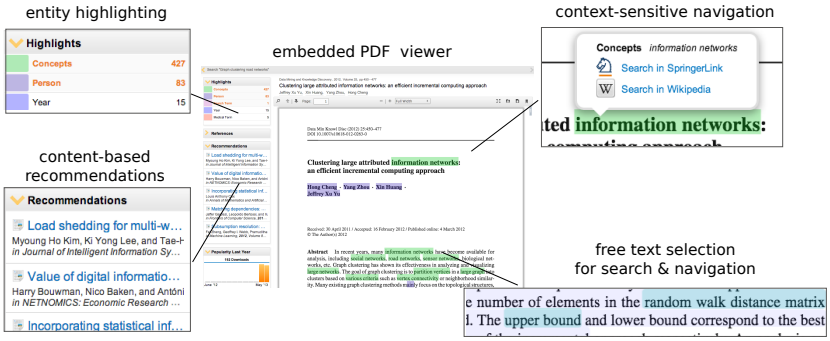

Fig. 4. Selected features for content navigation.

Visitors can freely play with the web application to experience the performance and responsiveness of the system. The application connects via HTTP to a HANA server hosted by SAP.

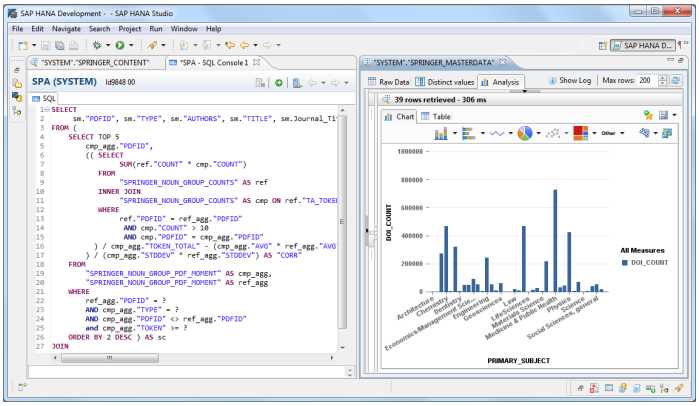

Fig. 5. SAP HANA Studio

Acknowledgements

The authors would like to thank the whole Strategic Projects Team SAP/Walldorf, especially Spyridon Antonopoulos, Jens Böning, Fredrick Chew, Enno Folkerts, Christian Heller, Nick Lanham, Andrew McCormick-Smith, Martin Sommer, Frederik Transier, and Patrick Zamzow. Furthermore, we thank Springer for their support.

References

1. Plattner H., Zeier A.: In-Memory Data Management: An Inflection Point for Enterprise Applications. Springer, Berlin (2011)
2. SpringerLink Corpus, `http://link.springer.com` (2013)
3. Wikipedia Encyclopedia API, `https://www.mediawiki.org/wiki/API` (2013)