Using a Knowledge Graph Data Structure to Analyze Text Documents (VAST Challenge 2014 MC1)

Florian Stoffel*
University of Konstanz

Fabian Fischer†
University of Konstanz

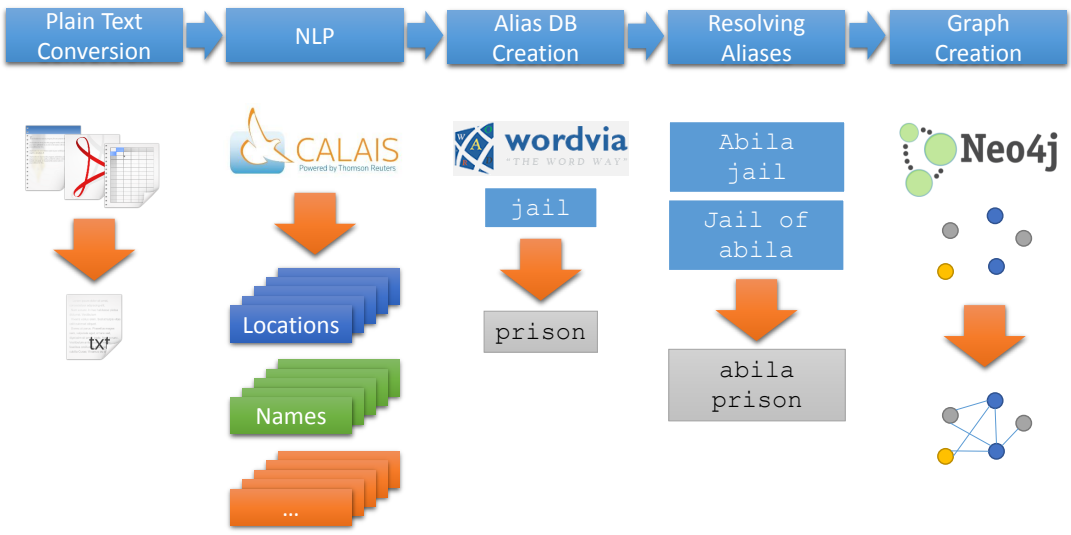

Figure 1: The data analysis and preparation pipeline. It covers the process of generating text data, extracting information, reduction of ambiguities (aliases), and graph creation.

Abstract

In this paper, we describe the tools and the analysis pipeline we used in order to create a data structure out of the given data. Afterwards, we introduce a workflow used to find answers to the questions of VAST Challenge 2014 MC1. Most of the processes presented in the following have been used solely to create, refine a graph (node-link diagram), which is the foundation for everything we did in this mini challenge.

1 Introduction

In this mini challenge the participants were asked to examine an incident happened during a fictive scenario in the island country of Kronos. Several employees of the fictive company GAStech went missing during a celebration in the company's headquarter. Given different documents, for example employee records of GAStech, documents covering the latest history of Kronos, and current and historic news, participants were asked to identify and characterize events and analyze the past and current structure of a group called POK (Protectors of Kronos).

Our submission is completely based on examining a graph, which contains all knowledge we were able to extract from the given data [1]. This knowledge graph worked reasonably well, and was able to include all the different data sources and properties contained in the data. We encoded relationships in time, for example, publication dates from news, important events from the employee cvs, and e-mail communication in the same data structure as relationships of persons or the relation of a location to different dates or events. This high degree of integration requires good data input, for example having the same date encoded differently (`2014-01-21` or `21-01-14`) can significantly reduce the utility of the graph.

*e-mail: Florian.Stoffel@uni-konstanz.de
†e-mail: Fabian.Fischer@uni-konstanz.de

IEEE Symposium on Visual Analytics Science and Technology 2014
November 9-14, Paris, France
978-1-4799-6227-3/14/$31.00 ©2014 IEEE

2 Tools and Libraries

In order to generate our knowledge graph, we use the following tools and libraries. We use **LibreOffice CLI** interface to convert the provided Microsoft Word documents into plaintext documents. **Calais** and **Apache OpenNLP** was used for processing of the text, which most notably includes entity extraction. **Wordvia** was used in order to be able to reduce the number of different entities and to correct spelling errors, we use the Wordvia dictionary to resolve ambigous terms. Our analysis and the answers to the given questions are based completely on a graph created from the given data. In order to be able to store and query the graph efficiently, we make use of **Neo4J**, which also provides a flexible query language (Cypher). **Gephi** was used to visualize sub graphs generated by graph queries and includes further filtering and analysis methods, which are useful in order to examine a graph in detail. **Elasticsearch** was facilitated for full text search and to find evidence of hypothesis. The automatic parts of the data analysis preparation pipeline are implemented using **Java**.

3 Data Analysis and Preparation Pipeline

This submission has a strong focus on the data cleansing and preparation process, because the quality of the resulting knowledge graph

and the ability to find facts, evidence, and hypotheses validation strongly depends on the quality of the input data.

The process of data cleansing and processing is an iterative and time consuming process. We formalized this process in a pipeline as shown in Figure 1. This formalization enabled us to execute improved versions of single steps, which reduced the overall time needed to process the data.

The most basic step is the transformation of all text data in the same, plain text format. Plain text data is easy to process, and not at last no special tools are required to read it. After having the data in plain text, we used natural language processing to extract entities, for example locations, names, and dates out of the data. Having a number of different entities extracted from the text, a custom tool was used in order to resolve ambiguities, different spellings, and spelling errors to improve the data quality. During the analysis, the relationship for each of the terms, phrases, and entities to its originating document has been kept intact. This occurrence data was used in the last step to create the knowledge graph by connecting documents and co-occurring terms with each other.

4 Knowledge Graph

The core of our submission is the knowledge graph. For each of the extracted entities, a node is added to the graph which is labelled by the entity itself. Also, each source of entities, for example news articles or employee csvs, are added to the graph as node as well.

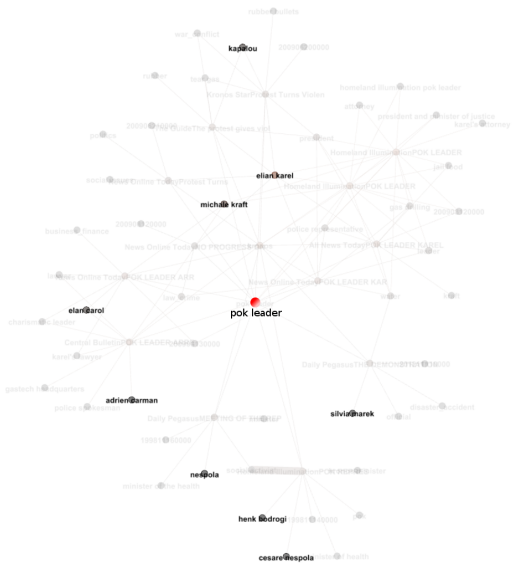

Figure 2: A subgraph originating at the *POK Leader* vertex. Persons are highlighted.

For each entity co-occurring with another entity on document level, we add an vertex with weight one from the entity to the document and the co-occurring entity as well. While adding new nodes to the graph, it is made sure that no node with the entity label already exists. If this is the case, the already existing node is re-used.

In order to take account of the different nature of entities (persons, dates, locations, etc.), each node has a set of *nature* properties, where we store the type of entities. For example, having a person named *Kronos* and an island *Kronos*, only one node would be added to the graph, but it will have a *person* and *geo-location* nature.

If a vertex to be inserted already exists, which for example can be the case for locations and dates, the weight of the vertex is increased by one.

The resulting graph is weighted and undirected, and allows a wide variety of queries. The following Cypher query finds all persons connected to the node labeled `POK` within a distance of 3: `match p=n--[*0..3]--m where m.label = pok and p.nature = person return n, p.length`.

The node link structure also gives us flexibility in terms of the returned type of data. There is no need to explicitly tell, that we are interested in email communication patterns from a person. Searching for nodes connected with others returns all available different types of data, because the graph has been constructed without making differences in the input data. Nevertheless, we could formulate restrictions in the query to only return nodes with specific data types.

5 Analysis Workflow

For making sense out of the data and to answer the questions from the challenge, we followed a four step workflow as shown in Figure 3.

The first step is the formulation and execution of a query on the knowledge graph. Secondly, the query results are inspected. Thirdly, we search for evidence of the analysis question in the query results. The last step, which is hard to separate from the second and third step is sense making, because during result inspection and evidence search, we automatically try to make sense out of the given data. Therefore, we didn't really separate the last two and sometimes the last three steps, but we kept the process formalized in terms of notes and annotated screenshots.

Figure 3: Analysis workflow. The arrow between the Sense Making and the Graph Query symbolizes a feedback, which makes the workflow suitable for repeated analysis on a specific subject.

Since additional knowledge can help to clarify and formulate the query of step one, it is possible to go from the sense making step to the query formulation step again. This makes sure, that we can narrow down a problem to it's cause, which is a process terminated by either finding evidence which makes sense, going back to reformulate and maybe also specify the graph query in more detail, or a point where the result is either empty or makes no sense.

The result of the analysis workflow is a documentation of all evidence, hypotheses, and additional knowledge which didn't fit in the analysis question, but could nevertheless be of interest for further analysis.

6 Conclusion

In this paper we showed our approach to Mini Challenge 1 from VAST Challenge 2014, which is highly data driven. A highly integrated knowledge graph unifying all provided data in a single data structure is the foundation for our submission. Also, the graph query language is invariant of data types or relationships, but can be specific if required by the analytical problem at hand. To provet he usefulness of our technique, we aim aim to combine all the different tools and libraries in a single visual analytics tool.

References

[1] J. J. Miller. Graph Database Applications and Concepts with Neo4j. In *Proceedings of the Southern Association for Information Systems Conference*, pages 141–147, 2013.