Reception of LGBT in Newspapers

Magdalena Bönisch Till Haubenreißer Maximilian Möller

February 23, 2017

Universität Leipzig, Introduction to Digital Humanities (Dr. Köntges)

Abstract Here are 150 to 200 words which constitute our abstract.

1 Introduction

This is our pretty Introduction.

2 Workflow & Implementation

In the following, the general workflow for analyzing the LGBT reception within newspapers is described. After that, the concrete decisions concerning the implementation are addressed.

Workflow The overall workflow is shown in Fig. 1. The upper part reflects the first step of collecting newspaper articles and creating a database. For each term from a predefined list of query terms an HTTP response to the API of an online newspaper archive is sent. In this work, the Article Search API of the New York Times[1] is used as it provides access to articles published since its foundation in 1851 and thus enables a time-dependent analysis. Furthermore, the returned JSON documents contain not only a URL to the actual article but also text snippets and lead paragraphs such that an analysis can be based on these text fields. Hence, no further call for the complete article is necessary. The API returns a JSON object consisting of meta-data like the number of hits as well as the actual articles. Since the access to the NYT API is limited per second and per day, the responses are parsed and stored into a MySQL[2] database. A relational database allows a quick access of the data along with SQL as a powerful query language for data selection and summarization. Thus, subsequent analysis could be based on

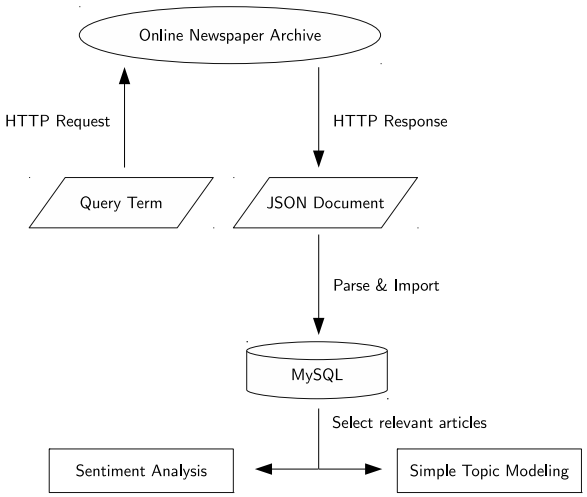

Figure 1: Workflow

the whole corpus with no dependency from the API. Furthermore, the proposed workflow is more flexible with respect to adding data from an additional newspaper archive since its response has only to be mapped to schema of the database and, especially, the analysis module (lower part of the workflow in Fig. 1) remains unaffected.

The analyses are based on two dimensions: query term and time. Thus, the input of the sentiment analysis and our simplified topic modeling approach is a set of paragraphs containing the query term and contained in articles published at a certain time. For instance, such an input can be all paragraphs from the database which contain the word *homosexual* and where published in the 1990s. The sentiment analysis assigned each sentence a vector of sentiment probabilities. We considered five sentiments: *very negative*, *negative*, *neutral*, *positive*, and *very positive*. The vector $\langle 0.2, 0.5, 0.1, 0.1, 0.1 \rangle$, for instance, describes the case that a sentence has a probability of 20% for being very negative, 50% for being negative, and 10% for being

[1] `https://developer.nytimes.com/`
[2] `https://www.mysql.com/`

neutral, positive, and very positive, respectively. For topic modeling, the joint occurrence of a target term together with another word (cooccurrence) is analyzed. Target terms are LGBT-related words, like the query terms. Each cooccurrence is assigned its frequency and whether its occurrence is significant. For instance, the cooccurrence $c_{2010s} = \langle \text{bisexual}, \text{rights}, 31, \text{true} \rangle$ means that within the paragraphs from 2010 to 2019, *bisexual* and *rights* occurred 31-times in the same paragraph and that this occurrence is significant. By clustering significant cooccurrences, the topics were manually created in order to yield high-quality topics.

Implementation The whole workflow is implemented in Java as a Maven[3] project. The project is managed on GitHub.[4] It consists of one module for collecting and one for analyzing the data. The database is accessed by using the Java Database Connectivity (JDBC) API. The used query terms are *bisexual*, *gay community*, *homosexual*, *lesbian*, *transgender*, and *transsexual*. However, the term *gay* was not considered as a query term because it appeared too often in non-LGBT-related contexts like as a given name or surname. In order to obey the time limit of the NYT API, only one request per second was sent. Almost all requests (95.4%) were successful. They returned a non-empty JSON object which could be inserted into the database. Failures of requests were due to denied accessed to the archive (HTTP 403) or gateway timeouts (HTTP 504). Because such failures were very rare, failed requests were not sent for a second time. All in all, the built corpus consists of $44,485$ articles whereby $93,7\%$ of them contain a non-empty lead paragraph on which the analyses are based.

The database models the mapping of query terms and keywords to articles. Keywords represent interesting additional information to the articles like associated persons, organizations, or geographical information. Approximately, 75% of all articles have at least one keyword. Moreover, an article has a URL which leads to its HTML representation in two-thirds of the cases; otherwise the article is only accessible as a PDF document.[5] Further attributes are the publication date, the actual text type (for instance, article, interview, or biography) and headline, abstract, lead paragraph, and snippet. Since the headline is very short, the abstract is missing for 62% of the articles, and the snippet is mostly the same as the lead paragraph, the analysis is based on the lead paragraph.

Essentially, the analysis module of the Maven project comprises the sentiment and the topic model package. They both rely on the sentence extraction task. For a certain search word and a certain publication date, this task selects windows of sentences from the paragraphs in the database. The window size is parameterized, i.e., Both of them depend on the natural language processing library Stanford CoreNLP[6]. This library contains annotators for tokenization, sentence splitting, part-of-speech tagging, and sentiment analysis.

//TODO: keyword analysis

3 Underlying Data

4 Results

4.1 Sentiment Analysis

This is our spectacular sentimental analysis.

4.2 Simple Topic Modeling

This is our great topic modeling.

5 Discussion

This is our insightful discussion.

5.1 Implementation Issues

5.2 Content Issues

5.3 Future Work

6 Conclusion

This is our awesome conclusion.

[3] `https://maven.apache.org/`
[4] https://github.com/macksimiljan/lgbt-news
[5] Older articles are archived as PDF not as HTML.
[6] `http://stanfordnlp.github.io/CoreNLP/`