ConTEXT Explorer: a web-based text analysis tool for exploring and visualizing concepts across time

Ziying Yang[1]**, Gosia Mikolajczak**[1]**, and Andrew Turpin**[1]

1 University of Melbourne

Summary

ConTEXT Explorer is an open Web-based system that assists in exploring the context of concepts (combinations of co-occurring words and phrases) over time in text documents. It provides a user-friendly interface for the analysis of user-provided text data and integrates functionalities of the Whoosh search engine, Spacy, Gensim, and Plotly Python libraries. By providing suggestions for query expansion and producing interactive plots, `ConTEXT Explorer` facilitates exploratory data analysis, which can serve as the basis for subsequent text classification.

Statement of need

With the availability of digital sources of data and associated tools, automated text analysis is becoming increasingly popular in the humanities and social sciences. While for very large corpora, unsupervised text mining methods like topic modelling (Blei et al., 2003) can provide some useful summaries of data, many social science and humanities applications require analysis of data in context. That is, simple "bags of words" automatically mined and presented in isolation from the original text are often not meaningful for complex questions involving human behaviour and society. Inevitably, human interpretation is required to make sense of such patterns. For corpora with more than several hundred documents, there is a need for computational tools that can assist researchers in exploring the context in which "bags of words" (we will call them concepts from now on) occurs.

Similarly, there is a need for tools that assist in the manual construction of concepts from text corpora. In particular, manual intervention to judge the semantic intent of words (e.g., word sense disambiguation) is usually needed to filter keywords to add to concepts that might be generated by automatic methods such as query expansion (Buckley et al., 1994) or comparison of word embeddings (Mikolov et al., 2013). For example, if a researcher is interested in finding articles about same sex marriage, they might start with "same_sex marriage"[1] as a concept. Automated methods processing a corpus of news articles might suggest related words like 'matrimony,' 'union,' 'erosion,' and 'puzzlement'[2]. Depending on the research question and the context of these words, some might be relevant to the concept and should be included, while others are not. It requires complex human judgement to make the distinction. `ConTEXT Explorer` is a tool to assist the construction of such concepts in context.

Most existing computational methods underlying automated text processing require at least a working knowledge of relevant methods and programming languages (such as R or Python).

---

[1]We use underscore to join multiple words into a single phrase.

[2]This is an example where we apply `ConTEXT Explorer` in the Australian Research Council Discovery Project (DP180101711) "Understanding Political Debate and Policy Decisions Using Big Data."

ConTEXT Explorer is designed to lower these barriers to entry, particularly for humanities and social science researchers, by allowing an application of information retrieval and machine learning methods to text analysis without programming knowledge.

Comparison with other tools

Current text analysis tools require either previous knowledge of programming (e.g., R, Python), or are commercial products (e.g., **RapidMiner** (RapidMiner, 2021), **Google Cloud Natural Language API** (Google, 2021)). One exception that we are aware of is **Voyant Tools** (Sinclair & Voyant Tools Team, 2012), which is an open-source web-based text analysis tool built in Java. It allows the users to explore their data using some basic text analysis techniques such as word frequencies (at the document level), word cloud, and word context (words appearing around a chosen term). ConTEXT Explorer provides several functionalities that give a user a deeper understanding of the text, which are currently not available in Voyant Tools, such as concept suggestions, sentence ranking and concept grouping. It includes models allowing discovery of similar terms, and a search engine allowing retrieval of sentences relevant to concept terms, which can be used for concept expansion, and visualization of concepts over time. One key feature is that a concept can be either conjunction or disjunction of bags of words.

Compared to commercial text analysis systems such as **RapidMiner** (RapidMiner, 2021), which include some complex analysis techniques, ConTEXT Explorer is open-source (free) and easy to install. It enables researchers to browse text interactively for concepts (bags of words) in their corpus before mining the text in machine learning-driven systems.

ConTEXT Explorer is designed to help users interested in defining concepts, and exploring their trends over time. This could be particularly helpful as an input for some popular text analysis systems such as **MonkeyLearn** (MonkeyLearn, 2021), which enable text classification, tagging, and training AI machine learning models but require prior knowledge of the data.

ConTEXT Explorer is engineered to allow for the integration of other Python packages into the analysis process. It can be easily combined with other Python APIs (such as MonkeyLearn), once the concept groups are defined.

Key features

ConTEXT Explorer is developed using **Dash** (Plotly Technologies Inc., 2015) in Python, and integrates the following packages.

- **Spacy** pipeline (Honnibal et al., 2020) - for pre-processing the text corpora uploaded by users.
- **Whoosh** (Whoosh, 2021) - for building a search engine, which allows ranking of sentences relevant to the given concepts, and word frequency analysis at the sentence and document level.
- **Gensim** (Řehůřek & Sojka, 2010) - for training a word2vec (Mikolov et al., 2013) model for the uploaded corpus, which allows the user to find words related to other words for expanding concepts.
- **Plotly** (Plotly Technologies Inc., 2015) - for visualizing results in interactive graphs, which can be customized and saved as PNG files.

ConTEXT Explorer has been tested locally under macOS and as a server running under Ubuntu and continuous integration tests are performed using Travis CI.

Build a corpus

Users are asked to format their text documents as a CSV file (with each document saved in a separate row), before uploading this file into `ConTEXT Explorer`. At a minimum, users are asked to provide document text and publication year. Users can also upload columns with additional document information (such as document author, title, and so on).

`ConTEXT Explorer` processes the submitted file in the following steps.

1. Sentenize and tokenize English text using Spacy (Honnibal et al., 2020). This allows ranking of documents and speeds up document search.
2. Index the documents, and build a search engine for the corpus using Whoosh (Whoosh, 2021) and the Okapi BM25F (Robertson & Zaragoza, 2009) ranking function.
3. Remove stop words, lemmatize remaining words, and generate a word2vec (Mikolov et al., 2013) model for the corpus using Gensim (Řehůřek & Sojka, 2010).

For each corpus, users can create a new analysis, or load a pre-saved analysis to the dashboard.
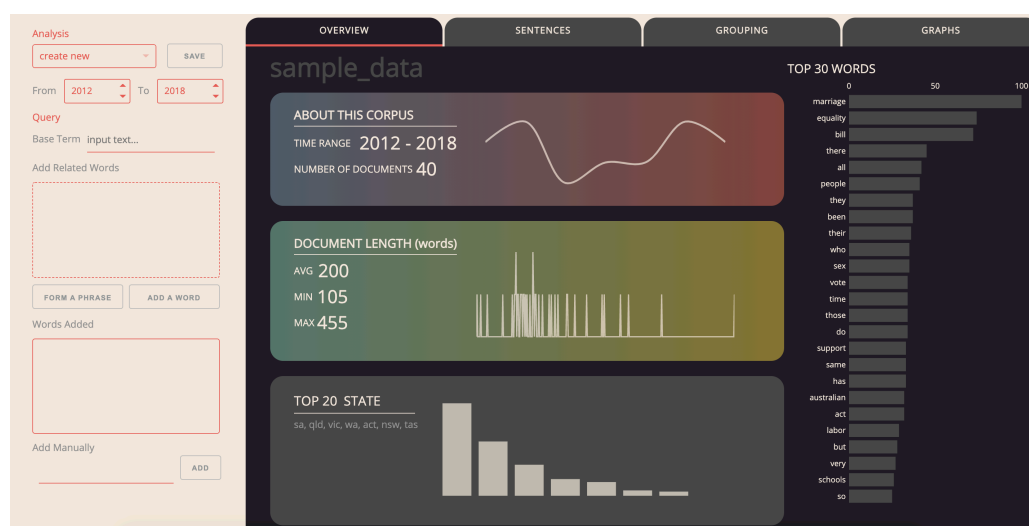
Dashboard

Figure 1: The starting dashboard.

As shown in Figure 1, the dashboard interface has two panes. On the left-hand side, users can:

- select the year range of documents to be displayed in search results;
- add or delete query terms (single words or phrases) to create a concept;
- save the current query as a new analysis; and
- download the subset of the corpus filtered by the query terms.

Overview. The overview tab summarizes the corpus information such as the total number of documents, year range, document length, most frequent words in the corpus, and most frequent values for selected metadata.
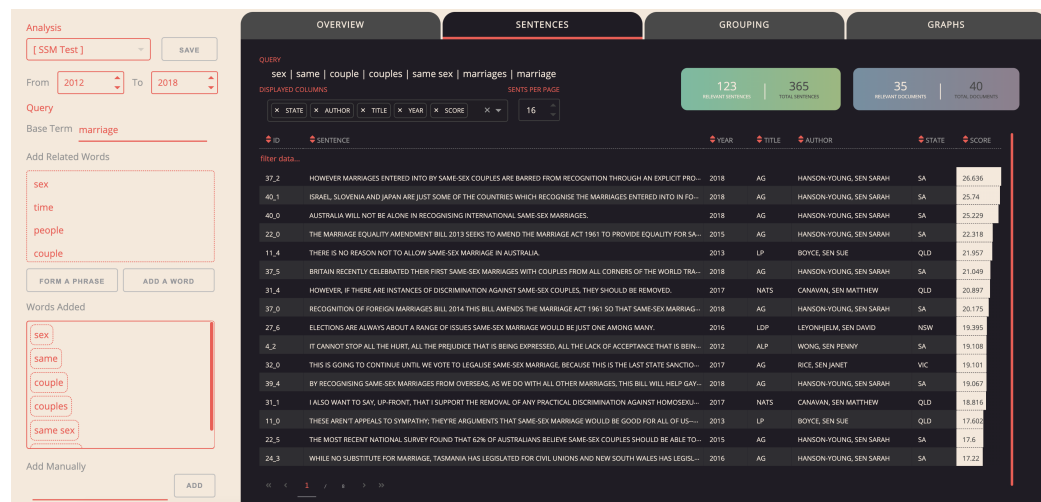
Figure 2: The sentences tab of the dashboard, with some query terms shown in the left pane.

Sentences. This tab shows the ranking of relevant sentences based on query terms defined in the left pane. Sentences are ranked by the Okapi BM25F ranking function, and the computed similarity score for each sentence is shown in the "SCORE" column. The table can be sorted and filtered by column values. Users can click on each sentence to see its full content in a pop-up window, which also allows checking of the frequency of individual terms and adding them to the query.

Figure 3: The grouping tab of the dashboard, showing the term frequency of the added terms across time (top), and some examples of query groups (bottom).

Grouping. The top part of this tab shows the number of sentences containing each query term within the user-defined year range. In the bottom part, users can group the query terms using "Any" or "All" operators. Groups can be further combined into more complex groups.
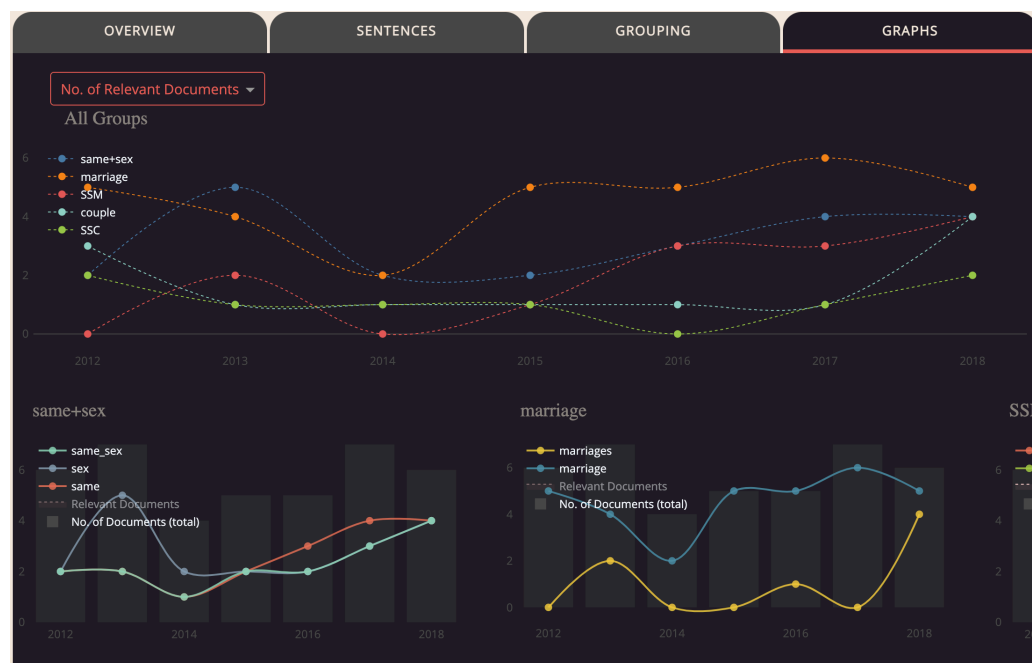
Figure 4: 'Graphs' tab, showing the aggregated graph for all groups (top) and individual graph for each group (bottom).

Graphs. Based on the query groups generated in the previous tab, this page displays aggregated and individual plots, which allow comparing groups (top) and individual terms within each group (bottom). Users can choose the number of relevant documents, the number of sentences, or the proportion of documents as the y-axis of the graphs. All graphs are plotted by Plotly (Plotly Technologies Inc., 2015) which allows users to interact with every trace in the graphs.

Save and reload an analysis

As mentioned in the section above, users are able to save the details of their analysis (including added terms and generated groups) and reload it to view all of the ranking, groups and graphs from the index page.

Acknowledgements

References

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *The Journal of Machine Learning Research*, *3*, 993–1022.

Buckley, C., Salton, G., & Allan, J. (1994). The effect of adding relevance information in a relevance feedback environment. *SIGIR '94*, 292–300. https://doi.org/10.1007/978-1-4471-2099-5_30

Google. (2021). *Cloud natural language – Google Cloud.* https://cloud.google.com/natural-language/

Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). *spaCy: Industrial-strength natural language processing in Python.* Zenodo. https://doi.org/10.5281/zenodo.1212303

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings.* http://arxiv.org/abs/1301.3781

MonkeyLearn. (2021). *How MonkeyLearn works.* https://monkeylearn.com/how-it-works/

Plotly Technologies Inc. (2015). *Collaborative data science.* Plotly Technologies Inc. https://plot.ly

RapidMiner. (2021). *Best data science & machine learning platform.* https://rapidminer.com/

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, *3*(4), 333–389. https://doi.org/10.1561/1500000019

Řehůřek, R., & Sojka, P. (2010). Software framework for topic modelling with large corpora. *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50.

Sinclair, G. R., Stéfan, & Voyant Tools Team, the. (2012). *Voyant tools (web application).* https://voyant-tools.org/

Whoosh. (2021). *Whoosh, a pure Python search engine library.* https://whoosh.readthedocs.io/en/latest/intro.html