Comparing STM and LDA for Topic Modelling in R

Steven Hernandez
University of Illinois, Urbana-Champagne
CS410, Winter 20211
stevenh5@illinois.edu

I. The Structrual Topic Model

[1]There is a kind of probabilistic topic model recently described which has gotten attention for its potential benefits to social science research, the *Structural Topic Model* (STM). Built on top of other generative models like *Latent Dirichlet Allocation* (LDA) and *Correlated Topic Model* (CTM), the Structural Topic Model enhances its topic modelling capabilities by allowing researchers to incorporate document metadata into their models. This metadata, which can *be* any numerical or categorical data associated to the documents, can modify the *topical prevalence* (the likelihood distribution of topics), *topical content* (the term distribution for the topics) or both.

[2]If we consider the LDA model, the likelihood distribution of topics and the term distributions of each topic are fixed. This is not ideal for certain tasks, such as social science research. Consider the task of performing topic analysis on U.S. Presidential Debates; here it is likely the frequency of specific terms such as the name of a politician or a policy term may vary depending on the ideological bias of the document source or of the particular date the document was produced. We would like to capture in some way how our knowledge of the world and of the document impacts *what* is spoken about and *how* it spoken about. These effects are precisely what the STM aims to capture through metadata, and we can see the potential benefit of the model is high when the documents we are analyzing have associated facts which are highly discriminating (as is the case with political text).

Having thus established the potential benefits of the STM, this paper aims to briefly explore the use of the model through its implementation in the `stm` library for the popular statistical programming language R. In Section II and III, the `stm` library is compared in its ease of use, functionality, and results to that of a typical LDA implementation in the same language. [3]The dataset being used for the demo is of a United Nations General Debate Corpus containing the statements of representatives from various member states from 1970 to 2016. Section IV summarizes the findings of this demo exploration.

II. Using the `stm` Library

[4]The `stm` library for R has its functionality divided into six domains. Usage begins by *ingesting* the documents and metadata. The data is *prepared* through preprocessing. Then, the STM model is *estimated* with the prepared data. Finally, the user can take the results of the estimation to *evaluate* them to select a model, *understand* the results by exploring the effects of the metadata on the distributions or see the ranking of terms per topic, and *visualize* the results by plotting for example the topic frequencies or creating a word cloud for a topic.

To begin with, the documents must be ingested and prepared before they can be fed to main `stm` function. This requires only a handful of lines of easily legible R script.

```r
library(stm)

# load data
setwd("~/Projects/CS410/tech_review")
data <- read.csv("data/un-general-debates.csv")
# preprocess text
processed <- textProcessor(data$text, metadata = data)
# associate text with metadata
out <- prepDocuments(processed$documents, processed$vocab, processed$meta)
```

Figure 1 Data preprocessing with the stm library in R.

The `textProcessor` function is capable of reading data in multiple formats. In this particular case, the documents were packaged in a *csv* file along with their constituent metadata. The function carries out an entire chain of sensible preprocessing steps on the raw text provided, including lowercasing, removing punctuation, stemming words and removing infrequent terms. All of these preprocessors can be adjusted through optional arguments to the function, making for a convenient and concise way to handle data ingestion. Compared to a similar implementation using popular R libraries DT and tm for a simple LDA, the difference is quite

clear. Not only does it take considerably more lines to write, but there is also quite a number of intermediary objects that need to be instantiated.

```r
library(DT)
library(tm)

# load data
setwd("~/Projects/CS410/tech_review")
textdata <- read.csv("data/un-general-debates.csv")
# load stopwords
english_stopwords <- readLines("https://slcladal.github.io/resources/stopwords_en.txt", encoding = "UTF-8")
# create corpus
corpus <- Corpus(VectorSource(textdata$text))
# preprocessing corpus
processedCorpus <- tm_map(corpus, content_transformer(tolower))
processedCorpus <- tm_map(processedCorpus, removeWords, english_stopwords)
processedCorpus <- tm_map(processedCorpus, removePunctuation, preserve_intra_word_dashes = TRUE)
processedCorpus <- tm_map(processedCorpus, removeNumbers)
processedCorpus <- tm_map(processedCorpus, stemDocument, language = "en")
processedCorpus <- tm_map(processedCorpus, stripWhitespace)
# compute document term matrix
min_frequency <- 5
DTM <- DocumentTermMatrix(processedCorpus, control = list(bounds = list( global = c(min_frequency, Inf))))
dim(DTM)
# removing docs with empty rows
sel_idx <- slam::row_sums(DTM) > 0
DTM <- DTM[sel_idx, ]
textdata <- textdata[sel_idx, ]
```

Figure 2 A possible implementation of data preprocessing using DT, tm in R.

After the data has been preprocessed, the `prepDocuments` function ensures the documents are properly associated with their metadata, and they can then be fed to the `stm` function to estimate the model. For this dataset, I used a topic size $K = 20$, a maximum of 200 EM iterations, and *spectral* initialization for both the STM and LDA implementations. For the STM implementation, the country of the speaker and the year of the debate were used as the metadata for topic prevalence, while only the year was used for topic content. The latter is due to a current limitation with the `stm` library which allows for only a single variable in the topic content formula. The STM estimation took longer to complete than that of LDA, despite converging after only 42 iterations (the LDA estimation ran for all 200 iterations), though only a matter of minutes for a dataset of over 5000 documents.

III. Comparing the Results

Once the models have been estimated, evaluating the results with the functions built into the `stm` library like the plot function was simple, efficient, and not dissimilar from what is available in other libraries such as `topicmodels`. [5]I used a code snippet from Martin Schweinberger's LDA tutorial to visualize the topic distributions of 5 randomly sampled documents. Included in the Appendix are the resulting distributions with the topic names composed of the top terms by score as measured by both models. The sample documents demonstrate one of the two notable improvements of the STM's performance over that of the LDA model. Namely, we can see that documents tended to score highly in fewer topics with STM over LDA, which makes interpreting the results more straightforward and classifying the documents easier. Also improved are the topic terms selected by the STM. Whereas the LDA model tended to prefer more general terms like "region", "develop", "nation", and often repeated these terms across multiple topics, the STM selected more discriminating terms such as the names of individual countries ("paraguay", "nepal", "andorra") and generally had fewer overlapping terms across topics.

IV. Conclusion

The ability to augment topic analysis with document metadata (STM) seems poised to advance social studies research beyond what has been previously capable with popular topic models such as LDA and CTM. Thanks to well designed, efficient, and user friendly `stm` library in R, this state-of-the-art technology is easily accessible to researchers as well as students. Having used the library to quickly demo the capabilities of the STM with political text, I found it to perform exceedingly well. It cut down on boilerplate often found in other popular topic modelling libraries, providing sensible defaults with the possibility to fine tune the model to meet the particular needs of task at hand. Sticking mostly to the default settings, I was able to get results that were markedly improved in terms of topic distribution and term distribution within topics for only a minor penalty in overall performance speed. While the library still needs more time to fully mature (the limitation to 1 variable for topic content being particularly notable), it is already a great tool.

APPENDIX

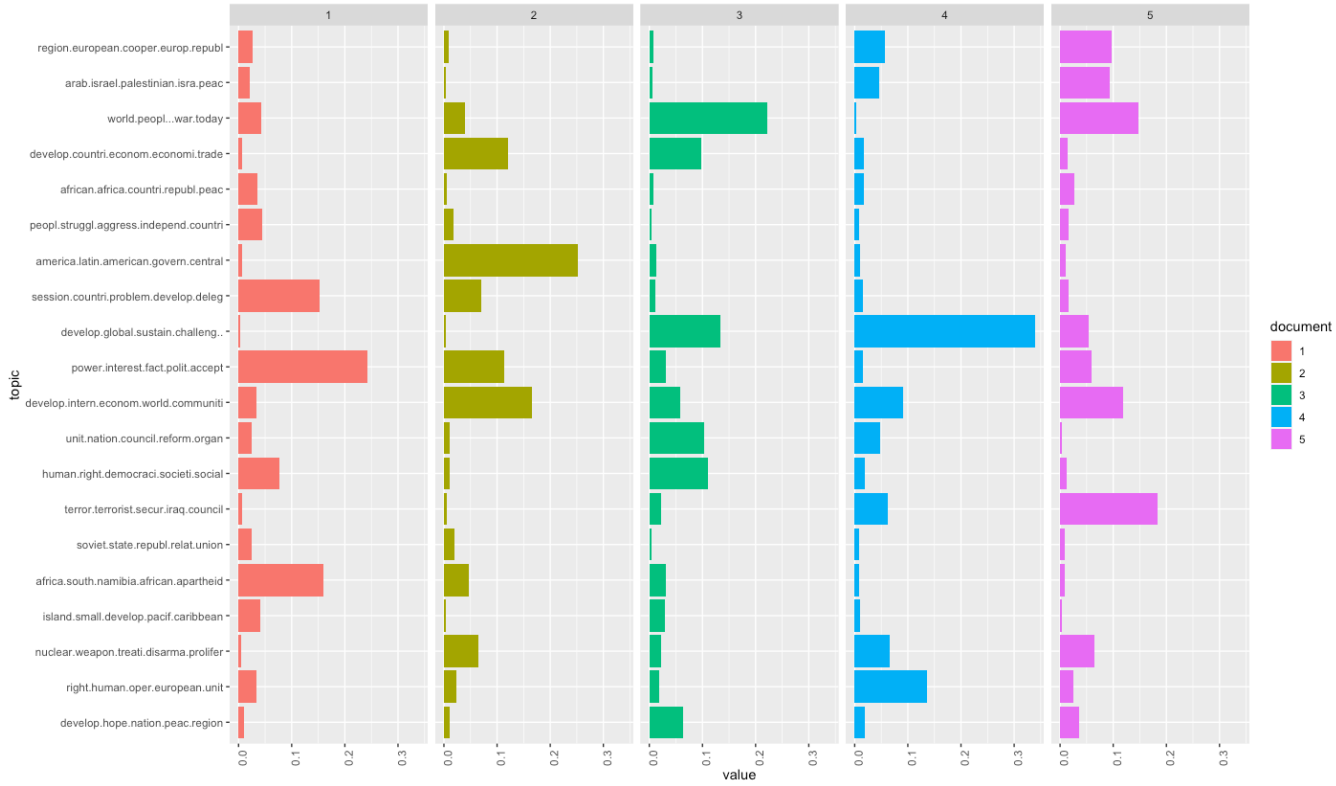

Figure 3 Topic distributions of sample documents using LDA

Figure 4 Topic distributions of sample documents using STM

REFERENCES

[1] Roberts, Margaret E., Brandon M. Stewart, Dustin Tingley, and Edoardo M. Airoldi. "The structural topic model and applied social science." In *Advances in neural information processing systems workshop on topic models: computation, application, and evaluation*, vol. 4, pp. 1-20. 2013.

[2] Lebryk, Theo. "Introduction to The Structural Topic Model," Towards Data Science (blog), April 17, 2021, https://towardsdatascience.com/introduction-to-the-structural-topic-model-stm-34ec4bd5383.

[3] Baturo, Alexander, Niheer Dasandi, and Slava J. Mikhaylov. "Understanding state preferences with text as data: Introducing the UN General Debate corpus." *Research & Politics* 4, no. 2 (2017): 2053168017712821.

[4] Roberts, Margaret E., Brandon M. Stewart, and Dustin Tingley. 2019. "Stm: An R Package for Structural Topic Models". *Journal of Statistical Software* 91 (2):1-40. https://doi.org/10.18637/jss.v091.i02.

[5] Schweinberger, Martin. "Topic Modeling with R," Language Technology and Data Analysis Laboratry (blog), October 01, 2021, https://slcladal.github.io/topicmodels.html#Topic_distributions.