Religious text – category mining

Documentation

Function of code

The main function of the code is to take religious texts and mine prevalent topics from them in order to form categories and sub-categories of the text. This is useful because it allows readers to find clusters of texts that are similar in meaning rather than having to search an entire religious book with a simple keyword search. Users can simply run the code and the code will print major categories of the text and list 10 sub-categories for each category. The code currently only mines a popular Islamic religious text called "Sahih Al-Bukhari", however the code can easily be extended to include any kind of religious text (although some custom pre-processing may have to be implemented depending on the format of the text) . The code uses domain-specific knowledge to refine the results, for example domain-specific stop words were added to the vectorization step. The code actually returns relatively good categories, and for the most part the subcategories are almost all related to the main category. Although the results are not perfect, I am happy with them!

Implementation

The first step that I implemented was data preprocessing. The original text is inputted as a .txt file, however there were many content-less lines throughout the file, so I simply used basic python strip() and replace() functions to get rid of meaningless and repetitive content such as volume number, book number, chapter number, and narrator name. I separated the text into documents according to paragraph. Since each paragraph in "Sahih Al-Bukhari" is independent from the previous and the next one – creating paragraph chunks, as documents seemed like the proper option. I then needed to represent these text segments in a way that the computer can deal with easily. So I used tf-idf vectorization – the same algorithm that we discussed in class. One important thing to note here is that I had to use my own stop word list in order to weed out certain highly repetitive domain-specific words that were ruining my results. I also set a maximum document frequency threshold to 2.5%. I made it that low because the text segments themselves are very short – each one being no longer than 250 words. So if a word is repeated more than 5 times or so, than I treat it like a domain-specific stop word and simply ignore that word from the vocabulary. After creating these vectorizations, I clustered them into 100 clusters using KMeans and Lloyds algorithm. Each cluster is supposed to represent a main category. After separating the documents into K categories, I topic-mined each category using a simple term-as-topic algorithm. However, instead of using word count, I used tf-idf scores. I think simply print out each main category and the top 10 topics for under every cluster, each representing a subcategory.

Running software

Running the software is easy. Make sure to run the code.py file using python 3.5 or above (earlier versions are untested). The following dependencies are required for the software to run: Sklearn & Numpy. Furthermore, make sure that the muslim_text.txt is in the same directory as code.py. Run the command:

```
python code.py
```

And the categories and subcategories should get printed.

Contributions

All code was written and conceptualized by Arken Ibrahim.