Project 4: Clustering r/AskWomen Posts By Content

Jessica Hudiono

Project design

Reddit is a popular forum divided into subcommunities by topic. However, the r/Ask* subreddits are generalist rather than specific, where the only theme is to ask questions about any topic. Often they are answered by a specific group like men or women or car salesmen. Some focus on learning more about the group asked, or they can serve as a jumping point for casual conversation.

I picked r/AskWomen because it has a high concentration of posts from women, which makes it unusual on male-dominated Reddit. According to their tagline, the r/AskWomen subreddit is intended to provide a female perspective on a wide variety of topics. By performing topic modeling on the r/AskWomen corpus, I hope to identify 1) What Redditors want to know about women and 2) What topics female Redditors posted the most about.

Data

For data, I downloaded the June 2017 collection, which was actually published in r/datasets. The dataset contained every post made in June 2017, including author, subreddit, entirety of the body, upvote/downvote score, whether it received gold, timestamp, and other miscellaneous fields. Overall the entire JSON file was about 40 GB.

r/AskWomen for that month saw 117k posts made by 15.5k unique authors. While I was doing EDA I found out that large datasets are risky business, I learned to use chunking in pretty much every step of my pipeline. I also wanted to filter out known bots and deleted posts. Removing those came out to 104k posts.

Tools

For data storage and initial experimentation, I used PostgreSQL and Pandas, and Matplotlib to help visualize patterns.

Libraries used in main workflow:

- NLTK: Stopwords corpus, lemmatizing.
- Gensim:
    - Simple preprocessing (converting to lowercase and tokenizing)
    - Ngram creation
    - Experimented topic modeling with LdaModel, HdpModel, LsiModel, TfidfModel
    - CoherenceModel for coherency metric

- Pickle: Saving results for later analysis.
- PyLDAvis: Graphics for analysis and presentation. This library was the main motivator for my choice of Gensim LDA over the other algorithms in Gensim or sklearn, since the interactive visuals made it faster to compare coherency of keyword groupings and dominance of different topics and keywords.
- Pandas: Result analysis

Metrics

I tried CoherenceModel as a metric, but was advised that it may not be totally robust. Ultimately I relied on manually verifying that topic keywords made sense, and sampling posts to see how well they matched their dominant topic.

After training the model, I went back to the original document collection and found the dominant topic for each document. Then for each topic T_n I sorted the posts to find the ones that were rated by the model as most strongly influenced by T_n. Most of the posts seemed to match their topic, but there were a few that didn't make sense.

Algorithm

I did a sort of "Grid Search", adjusting the following components/parameters.

Algorithm	Feature	Ngram	Number of Topics	Iterations	Chunksize
- **LDA** - LSI/SVD - HDP	- **Word count** - TFIDF	- **unigram** - bigram - trigram	3, 4, … **9**, 10, 11, 12, 15, 20, 25, 30, 31	- **25** - 50	- 500 - **1000**

I was not able to get HDP to return a good set of topics, but otherwise the output did not differ by much. Since TFIDF and bigrams/trigrams did not make a real improvement I decided to leave those out for faster processing, and I chose LDA for my presentation since I could interpret the results more efficiently using PyLDAvis.

Best number of topics was difficult to discern. I was still getting "hybrid" topics at 10+ clustering, where unrelated posts were grouped together (posts entirely about food + posts entirely about clothing). However increasing the number of topics did not separate them out well, instead more topics with "random/misc" keywords were produced. I settled with 9 topics as the best balance.

Conclusions

There were a few consistent patterns over many rounds of testing even with different parameters. First, LDA always produced a grouping of moderation-type comments, based on phrases like “your comment has been removed”. Significant moderator presence makes sense given r/AskWomen’s stated intent to create a “safe space” for women. Interestingly, performing topic modeling with the same model and parameters on r/AskMen did not result in a moderation cluster. I can’t tell if the difference is because r/AskWomen is targeted more often by trolls, or standards for acceptable content is more rigorous than r/AskMen.

Another common theme was that a “Mood & Thoughts”-type topic was usually the biggest group. This topic was characterized by keywords like “I’m”, “feel”, “want”, “things”.

Other common topics were entertainment recommendations, clothes and shopping, attractiveness and looks, social gossip, daily routines, and food.

Next time

I may be able to make more improvements with more parameter tuning and investigating different libraries more thoroughly, such as LdaMallet. Furthermore, I had some preprocessing and transformation difficulties with lemmatizing and stopwords. Many words were not stemmed correctly and NLTK’s default stopwords corpus was not strict enough with filler words. Imgur and other links in particular were confusing to my model and probably should have been removed in the preprocessing or transformation stage.

I also would like to do better feature engineering with regards to word frequency vs TFIDF. Reddit conversations mostly occur around threads about a specific topic, which can then be broken down further into posts and their replies. Instead of treating each individual post as a document, it may make more sense to use whole threads or parent and children comments as documents.

Finally, I think it could be interesting to pull in upvote/downvote score or text sentiment as features or target variables.