Reddit News Recommender

Chan, Natural

17 May, 2022

Abstract

The goal of this project is to create a web app that will recommend posts when given a new post. We will primarily observe the posts from the news and world news subreddits. From there, we will perform dimensionality reduction and identify key topics within the subreddits and then recommend posts that are related to a new post. Once we get a working model, we will create a web app using Flask and then deploy it with very simple functionality.

1 Design

Reddit has grown into a massive social media platform, and has become a major news source for many as well. With new posts constantly being submitted, it is hard to keep track of every single update. This leads to individuals missing context on newer posts when they may have missed a previous post that would have provided relevant information.

Using the titles of Reddit posts, we want to recommend other posts that are related to newer posts. We will achieve this by using matrix decomposition on a corpus of posts submitted to the news and world news subreddits. We will then create a web app that will allow a user to input the URL of a new Reddit post and then return several potentially reelated posts.

2 Data Ingestion/Storage

The PushShift API saves posts from Reddit, providing a quick way to retrieve posts compared to the Reddit API. We used PMAW, a wrapper for the PushShift API to gather all available posts from the news and world news subreddit starting from January of 2010 to the present. Unfortunately, the PushShift API does not update information on posts after retrieval. This means that many posts will permanently be recorded with 0-2 upvotes (score) which would result in many erroneous posts in our data. To work around this, we obtained the ID for each post and used PRAW, a wrapper for the Reddit API, to retrieve the upvotes of 100 posts at a time in order to filter out any post with under 1000 upvotes. Note that due to PushShift API hardware problems, many older posts are missing so the data is skewed to more recent years. We will need to constantly pull new posts from the APIs and store them so we don't

end up missing out on any posts that would be lost in the future. Using SQLAlchemy, we stored all the posts into a database for easy access later.

3 Processing

3.1 Data Cleaning/Preprocessing

After several attempts at tokenization, we created a custom tokenizer using SpaCy which lemmatized each word and handled symbols/non-ascii punctuation, non-nouns/non-verbs, and any erroneous patterns.

3.2 Modeling

We primarily used matrix decomposition in order to perform topic modeling. Specifically, we used Non-negative Matrix Factorization for several reasons. NMF provides more interpretable results than LSA and is more effective on short documents compared to LDA. For our final model, we modeled for 20 topics which did a fairly good job at explaining the variance between posts.

In order to use the model and DTM for recommendations, we pickled the vectorizer and NMF objects using cloudpickle to handle any UDF dependencies and stored the DTM into our database for later.

4 Deployment

Once we had a properly functioning model, we wanted to create a web app using Flask. The web app is intended to have a very simple function - allow a user to input a Reddit URL and then recommend 5 posts. First, the web app requests an input from the user and then it calls a function that will apply obtain the title of the associated post. The function will then take that title and then do two key processes: apply the trained vectorizer and then apply the trained matrix factorization model on it. The end result is how much the post fits into each of the topics that we found from topic modeling. It will then use a similarity metric to determine which posts to recommend from the corpus. Since we saved the URLs of all of the posts in the corpus as well, we are able to share the title of the posts and attach a hyperlink to each. After getting the web app to function with gunicorn, we deployed it using Google Cloud.

5 Testing/Robustness

Recall that we want the data to be ingested constantly which also means we want the model to be retrained every time we get new data as well. To do this, we created a cron job that would automatically run our python scripts to access the API and then retrain and repickle the model everyday. We also wanted to limit what kind of user input was allowed. Luckily,

when given an input, we already had to access the submission using the Reddit API in order to get the title. When doing so, the Reddit API will automatically check if the input is a valid Reddit URL. This allowed us to validate the user input and return an error instead of crashing the web app.

6 Tools

- PRAW and PMAW to gather data from Reddit
- Pandas for data processing and cleaning
- NLTK and SpaCy for tokenizing corpus
- Sci-kit learn for vectorizing and modeling
- Matplotlib and WordCloud to generate plots
- SQL and SQLAlchemy for data storage
- Pickle and Cloudpickle to save and load models
- Flask for web app implementation
- Google Cloud for web app deployment
- cron to automatically access APIs and retrain models everyday

7 Communication

Presentation slides and visualizations are located in my github repository.