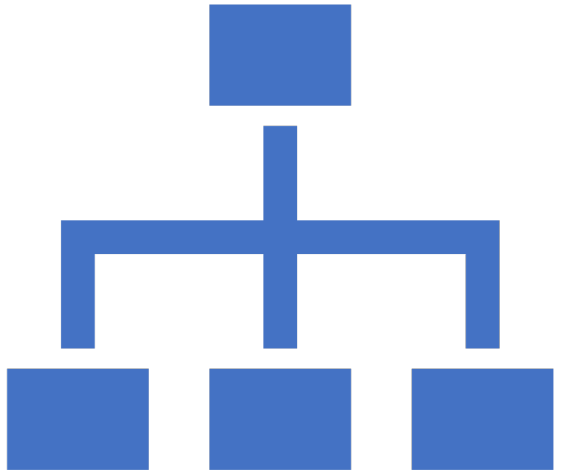
Reddit classification

Payal Chodha

10.18.2019



Problem

Reddit relies on users to organize their posts into sub-reddits, which are often mis-classified

Build a model to accurately reclassify the posts back to their subreddits.

Goal

Collect posts from two subreddits using Reddit's API

Use NLP to train a classifier on which subreddit a given post came from

Travel

- r/travel is a community about exploring the world.
- It has a lot of images and a lot of text data as well.

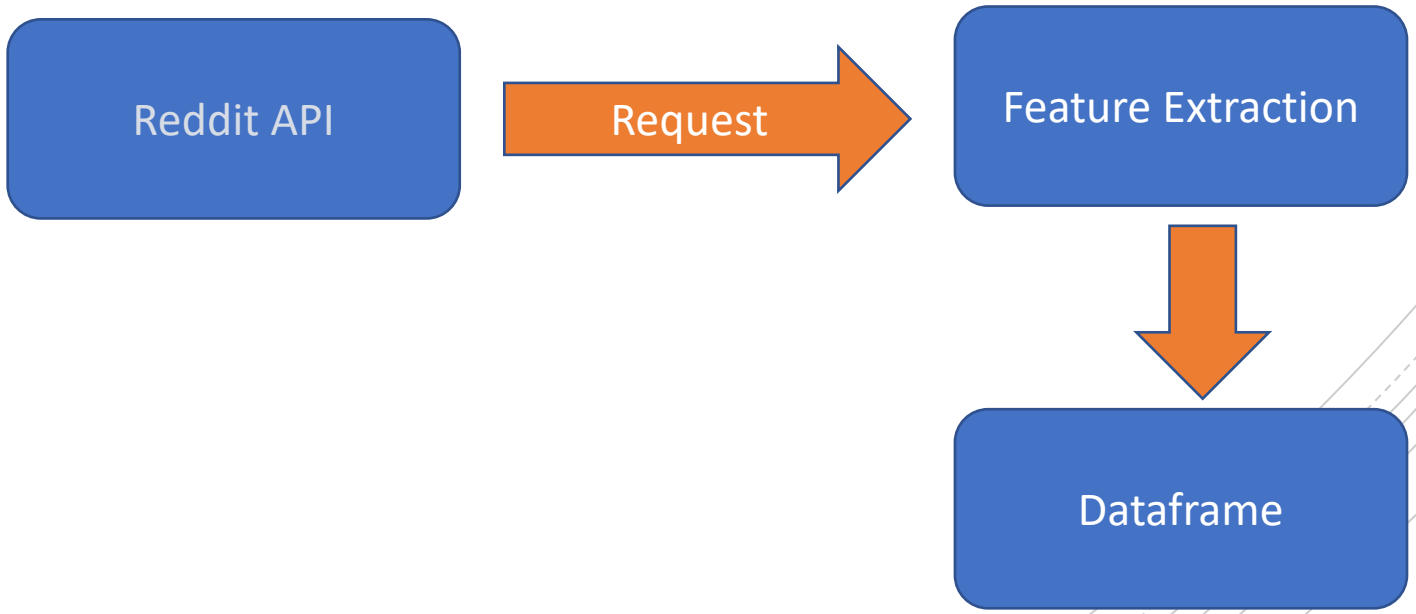
AwardTravel

- r/awardtravel is a place to discuss burning airline miles and hotel points.
- It has a lot of text data as well.

Data scraping:

The process is fairly simple:

- it’s just a basic request set up to access the API
- Managed to scrape about 15,000 posts in total



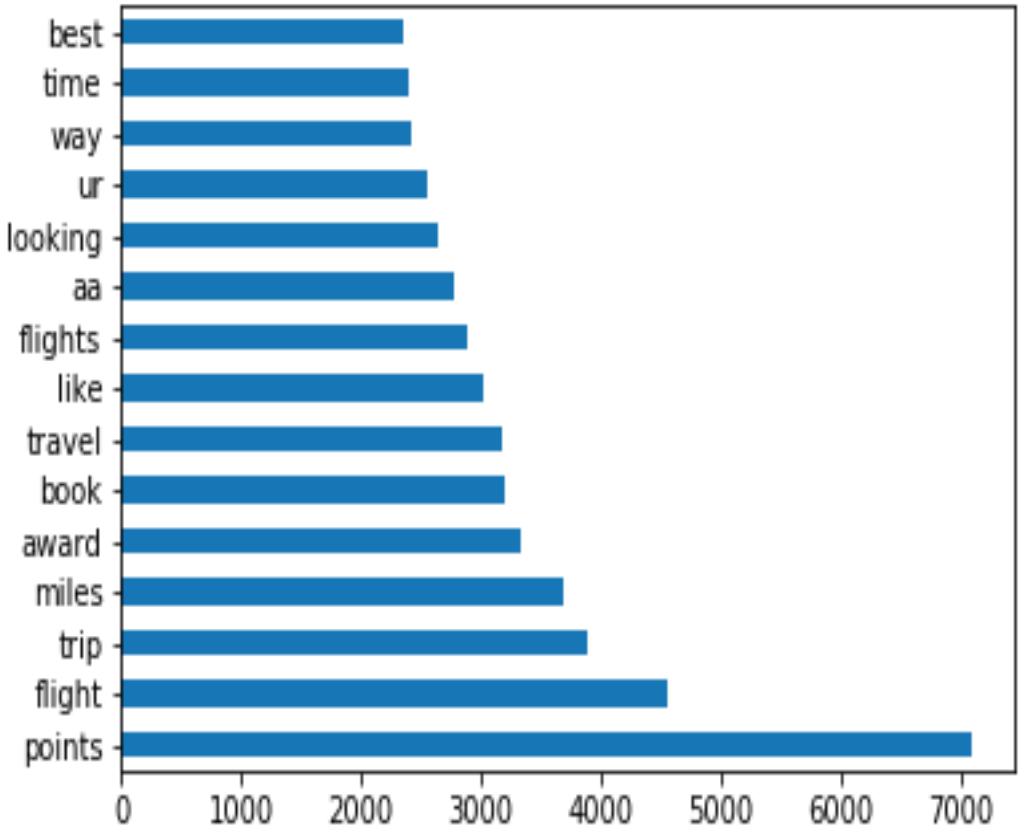
Data cleaning:

Cleaning up text is a lot easier than having to clean up numeric data:

- Remove nulls
- Filter non-letter characters, and URLs
- Filter emojis
- Convert the text to lower case

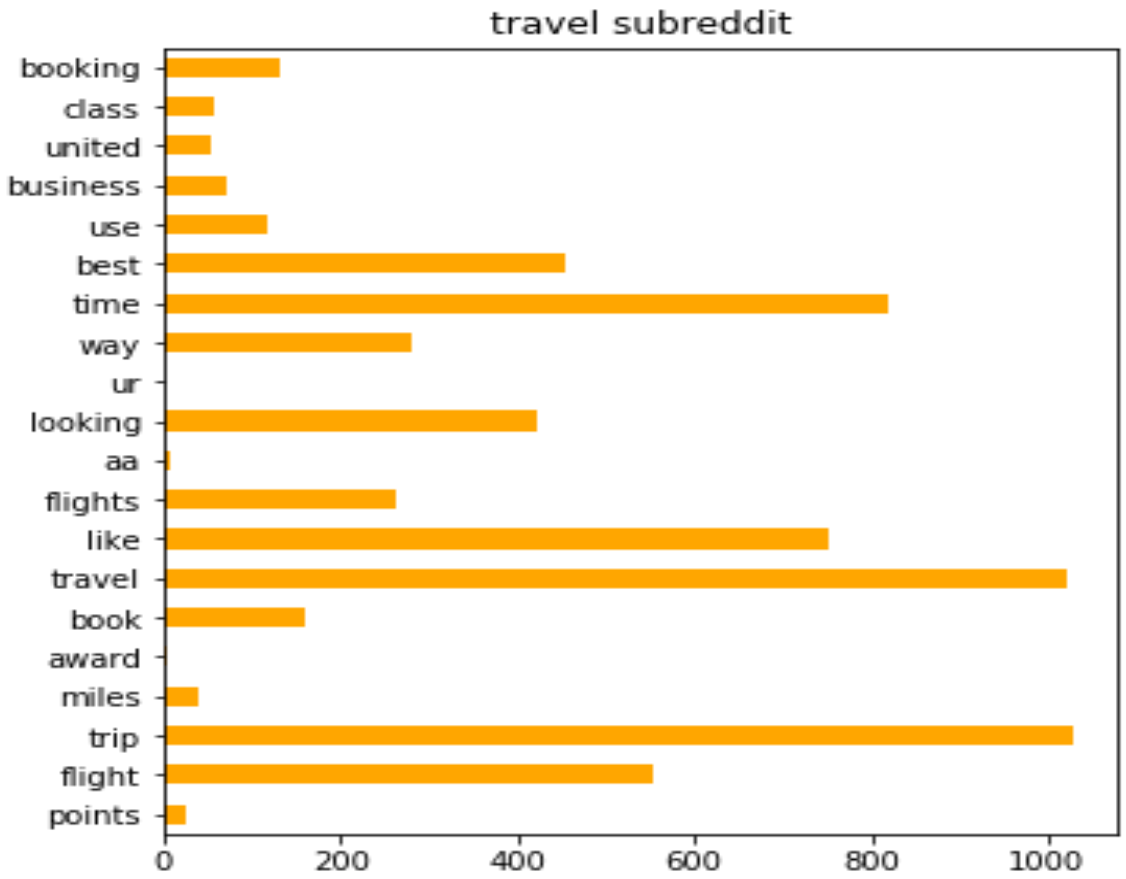
Data Preprocessing:

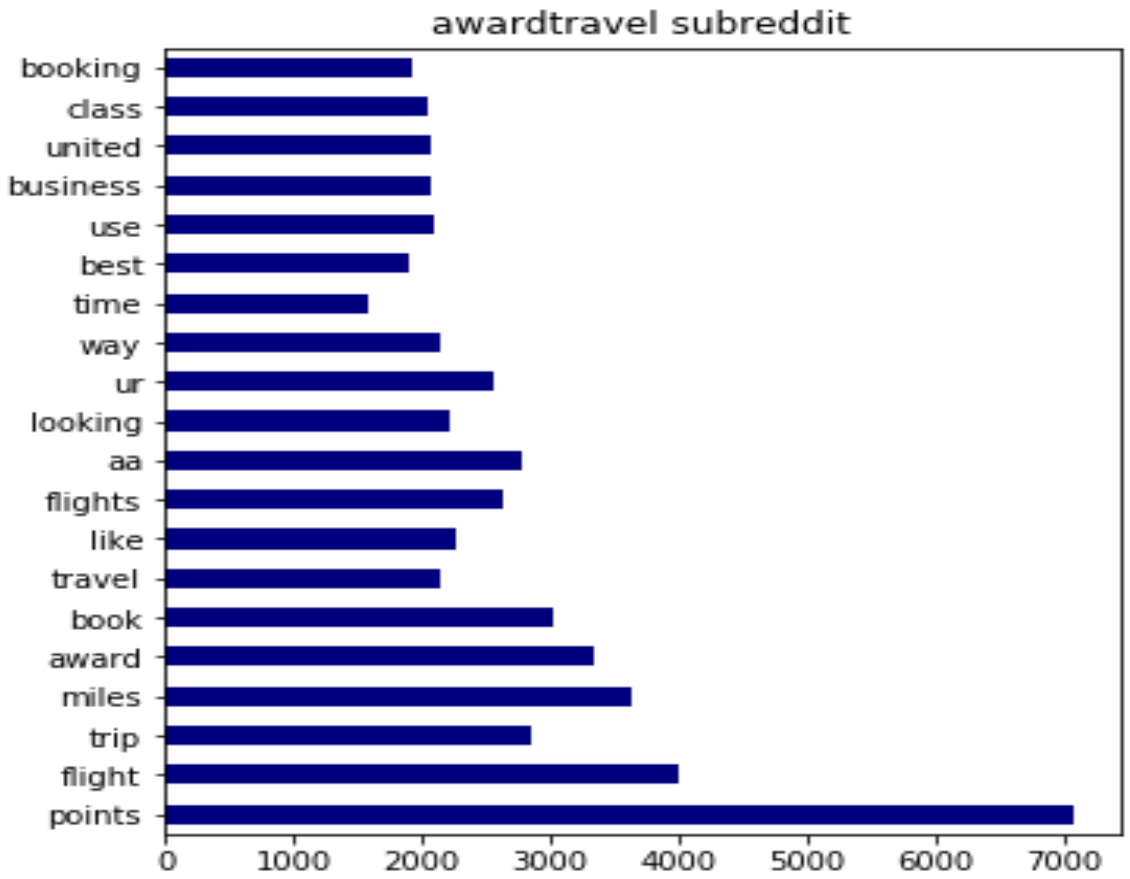
- Two sources of text in the posts: the title and body (the actual text in a post)
- Decided to combine the two sources into one feature, so that modeling would be a little bit easier.
- Looked for the common words.



Frequency of common words in both the subreddits

Added these common words to the list of stop words.





Lemmatizing or Stemming?

I chose lemmatizing over stemming because

- Lemmatizing is a more gentle process that seeks to return the dictionary form of a word
- Stemming reduces a word to its stem, which can return non-words

Model Selection:

- Each model was run twice: once with count vectorization and once with TFIDF vectorization
- Every model was optimized with grid-searching
- Models considered were:
    - Logistic Regression
    - Multinomial Naïve Bayes
    - Random Forest

Model Performance: Logistic Regression

CountVectorizer-	Logistic Regression:	
	Training score:	0.9684
	Testing score:	0.9333

TfidfVectorizer-	Logistic Regression:	
	Training score:	0.9558
	Testing score:	0.94051

Best parameters:
'cvec__max_features': 5000,
'cvec__ngram_range': (1, 3),
'cvec__stop_words': [stop]

Best parameters:
'tvec__max_features': 3000,
'tvec__ngram_range': (1, 3),
'tvec__stop_words': stop

Model Performance: Multinomial Bayes

CountVectorizer-	Multinomial Bayes:	
	Training score:	0.9178
	Testing score:	0.9018

TfidfVectorizer-	Multinomial Bayes:	
	Training score:	0.9212
	Testing score:	0.9087

Best parameters:
'clf__alpha': 0.01,
'cvec__max_features': 3000,
'cvec__ngram_range': (1, 2),
'cvec__stop_words': [stop]

Best parameters:
{'tvec__max_features': 3000,
'tvec__ngram_range': (1, 3),
'tvec__stop_words': None}

Model Performance: Random Forest

CountVectorizer-	Random Forest:	
	Training score:	0.8365
	Testing score:	0.8367

TfidfVectorizer-	Random Forest:	
	Training score:	0.8581
	Testing score:	0.8522

Best parameters:
'clf__alpha': 0.01,
'cvec__max_features': 3000,
'cvec__ngram_range': (1, 2),
'cvec__stop_words': [stop]

Best parameters:
{'tvec__max_features': 3000,
'tvec__ngram_range': (1, 3),
'tvec__stop_words': stop}

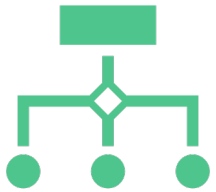
Future aspects and recommendations:

- Random forest has not very high accuracy scores for both Countvactorizer and Tfidfvectorizer,but but both of them doesn’t show high variance i.e. overfitting
- Logistic regression performs good with cvec and has even better accuracy score for Tfidf, and it also doesn’t show high variance.

So I recommend employing Logistic Regression with TfidfVectorizer moving forward.

Conclusions:

Able to predict with good accuracy the classification of a certain post



Can be leveraged by Reddit to ”suggest” sub-reddits when a user a writing a post

Can unlock ways to better target “award travel” customers for credit card offers