Utilizing Clustering Techniques to Generate Reddit Recommendations

Jonathan Wiseman

Elizabethtown College

(717)857-4069

wisemanj@etown.edu

**ABSTRACT**

Document clustering is an application of cluster analysis that operates on collections of text. Its aim is to produce a clustering such that documents in the same cluster are more similar to each other than to documents in other groups. Reddit is a popular content aggregation site that allows users to make and comment on “posts” in dedicated communities known as “subreddits.” Treating each subreddit as an aggregation of comment data allows the creation of documents that can be used in clustering. Document clusterings can be used to make personalized recommendations for active users that have defined interests. Additionally, subreddits can be visually represented based on their textual content, enabling a holistic view of Reddit’s communities.

# INTRODUCTION

## Reddit and PRAW

Reddit is a popular news aggregation and content hosting website that allows users to form directed communities, called subreddits, for discussion of relevant topics. The fifth most visited site in the United States, Reddit boasts upwards of 300 million unique users. Users can share content in a variety of ways: uploading of original content (including images and videos), links to content hosted elsewhere (such as news articles), and by making text-only “self-posts;” users are also able to comment on content. A positive-negative voting system, collectively referred to as karma and accumulated through “upvoting” and “downvoting” content, gives explicit feedback on comments and posts.

Reddit provides access to comment data to application developers through its API; access is granted to any Reddit user with a verified account. PRAW is a Python package that allows for automated scraping of comment data. The full documentation is available in [INSERT CITATION].

## Document Clustering

Cluster analysis is an unsupervised learning technique that seeks to group a set of objects together such that objects in the same group are more similar to each other than to objects in other groups. Document clustering is a problem in cluster analysis that seeks to cluster together “documents” based on their natural language content. In this context, documents may be web pages, books, news articles, etc. Document clustering is being used for a variety of tasks in information retrieval, including search engine optimization, topic extraction, fast information retrieval, and resolving lexical ambiguity [1].

Document clustering often relies on the representation of documents as word vectors; this is known as the bag-of-words approach. Bag-of-words entails representing each document as a sparse matrix consisting of binary attributes (with features being words: a 0 denotes the absence of a word, a 1 indicates the presence of a word) for each word in the collection’s vocabulary. To reduce the size of the vocabulary, stop words (common words like: a, the, is, etc.) are removed and some textual preprocessing (like lemmatization or stemming, both of which seek to get the root form of words) are performed [2].

Next, a measure of similarity or distance (called the metric) between word vectors must be chosen. Examples include Euclidean distance, cosine similarity, the Jaccard coefficient, Pearson’s correlation coefficient, and averaged Kullback-Leibler Divergence [2]. With document vectors and a sufficient similarity metric, a clustering algorithm must be chosen. Examples include k-means (in which documents are separated into k groups based on distance from a group’s centroid) and hierarchical clustering methods.

Additionally, other preprocessing can be performed to increase model performance, such as feature selection (further selection of important words), word weighting (ex. using tf-idf weights), and standardizing of vector lengths to 1 to avoid long documents dominating the analysis [2]. The typical bag-of-words approach yields thousands of features corresponding to each word in the corpus. Dimensionality reduction techniques, notably singular-value decomposition, can be used to reduce these matrices into more easily usable representations.

Finally, numeric-array representations of documents can be created using the Doc2Vec algorithm [INSERT CITATION]. Doc2Vec is trained on a corpus in order to “learn continuous distributed vector representations for pieces of text.” In most implementations, the final dimensionality can be specified prior to training, giving designers control over space and complexity.

# Design

Figure 1 displays the overall project outline:

A picture containing map

Description automatically generated

**Figure 1: Project Steps Overview**

## Approach

In order to utilize the external measures of completeness, homogeneity, and v-score on trained clustering models, eight subreddit categories were devised to serve as labels. For each category, four to five representative subreddits were determined for data scraping.

The project pipeline can be broadly divided into seven steps:

1. Scraping of raw comment data using PRAW
2. Cleaning of comment data
3. Document embedding training
4. Exploratory data analysis
5. Subreddit Clustering
6. Post Clustering
7. Application

Evaluation of clusters was performed using both internal and external measures. Silhouette score was used as an internal measure; completeness, homogeneity, and v-score were used as external measures.

The desired outcome of this project was to produce a recommendation system capable of recommending subreddits. To accomplish this, subreddits were clustered utilizing K-Means, Hierarchical Agglomerative, DBSCAN, OPTICS, Spectral, and Affinity Propagation clustering algorithms; individual posts were also clustered. To produce a recommendation for a post or subreddit, the other members of that object’s cluster are used. For this end, cluster size becomes a concern: clusters that are too large will produce an overwhelming number of recommendations. As such, maximizing homogeneity will produce clusters with a high degree of topic cohesion suitable for a recommender system.

## Data collection: comment scraping

Eight categories were established to provide ground-truth labels: art, gaming, music, politics and news, reading, science, sports, and technology. The breakdown of subreddits included in each category is given below:

1. Art
   1. r/art
   2. r/drawing
   3. r/painting
   4. r/photography
   5. r/pixelart
2. Gaming
   1. r/games
   2. r/gaming
   3. r/pcgaming
   4. r/ps4
   5. r/xboxone
3. Music
   1. r/guitar
   2. r/learnmusic
   3. r/musicology
   4. r/musictheory
   5. r/singing
4. Politics and news
   1. r/news
   2. r/PoliticalHumor
   3. r/politics
   4. r/worldnews
5. Reading
   1. r/bookclub
   2. r/books
   3. r/bookdiscussion
   4. r/currentlyreading
   5. r/suggestmeabook
6. Science
   1. r/biology
   2. r/chemistry
   3. r/physics
   4. r/psychology
   5. r/science
7. Sports
   1. r/cfb
   2. r/nba
   3. r/nfl
   4. r/soccer
8. Technology
   1. r/futurology
   2. r/gadgets
   3. r/linux
   4. r/tech
   5. r/technology

For each subreddit, the top first-and second-level comments from the top thirty most-upvoted posts were scraped. This totaled about 100 comments per post, for about 3000 comments for each subreddit. With thirty-seven subreddits represented, the total dataset included approximately 111,000 comments across eight categories.

## Data preprocessing

Raw comment data was processed to accomplish two goals:

1. Removal of non-alphabetical characters
2. Stemming and lemmatization

Reddit comments contain a number of characters that are not suitable for textual analysis. These include emojis, hyperlinks, hashtags, punctuation, tagging of other subreddits and users, numbers, and other miscellaneous characters.

Following removal of extraneous characters and text segments, stemming and lemmatization were performed to reduce words to their base forms. For stemming, both the Lancaster and Porter stemmers as implemented by Python’s NLTK were used. For lemmatization, NLTK’s WordNet lemmatizer was used. The results of each of these distinct processors were stored, so as to compare results.

Finally, stemming and lemmatization results were processed to remove “stubs.” Stubs are dangling apostrophes and endings that are kept after reduction. For example, the word “can’t” might be lemmatized to two fragments: “can” and “’t”; only the first result is relevant.

Performing document clustering with different stemming types does not yield significantly differing results. As such, all models were visualized and clustered with lemmatized text; additionally, using lemmatized text allows for the creation of meaningful word clouds (since stemming does not generally produce dictionary words).

## Training Document Embeddings

The Doc2Vec algorithm must be trained on a text dataset in order to produce document embeddings. The steps are as follows:

1. Create tokenized, labeled corpus
2. Train embeddings with dimensions 2-100
3. Find dimensionality that maximizes homogeneity with a K-Means model

The overall process of training document embeddings was performed very clumsily; more advanced knowledge of the Doc2Vec algorithm would be necessary for more informed training. As a metric for evaluating each dimensionality, a K-Means model was trained on the resulting embeddings; the silhouette, homogeneity, completeness scores and v-measure were reported.

Embeddings were trained with dimensionality ranging from two to one hundred, in steps of five for a total of twenty models. As dimensionality increased, silhouette score decreased dramatically. The range two to ten produced the best tradeoff of silhouette score to homogeneity. Amongst that range, a dimensionality of four produced the greatest homogeneity score. Without parameter tuning, the K-Means model trained on these document embeddings had homogeneity 0.77 and silhouette score 0.40.

## Exploratory Data Analysis

Exploratory data analysis can be divided into three steps:

1. Visualizing TF-IDF Vectorization
2. Visualizing Document Embeddings
3. Visualizing word clouds

TF-IDF vectorization is an extension of the standard bag-of-words approach to document vectorization: rather than using static counts of words for each document, each term’s term-document inverse-document frequency measure is used. Broadly speaking, the tf-idf is a measure of a word’s importance to a specific document. For example, the word “think” has high counts across many different subreddits (since it is often used to denote opinion); thus, its tf-idf score is low (since it holds no importance for any particular subreddit). The word “football,” however, is found in a relatively narrow span of documents (namely, a few of the sports subreddits). As such, it’s importance for those subreddits very high, and thus would receive a high tf-idf score.

For this step, both the TF-IDF and Count vectorizations were visualized. Figures 2 and 3 show the results of tf-idf and count vectorization, respectively:

A screenshot of a cell phone

Description automatically generated

**Figure 2: TF-IDF Vectorization**

**A screenshot of a cell phone

Description automatically generated**

**Figure 3: Count Vectorization**

Count vectorization tends to create large groupings, highlighting its inability to distinguish between which textual information is important for a subreddit’s clustering.

The n-grams used in vectorization were also altered. For both TF-IDF and count vectorization, 1-, 2-, and 3-gram vectorizations were explored. Overall, altering the dimensionality of the n-grams did not change the overall shape of the distribution. Figures 4 and 5 show the 2- and 3-gram tf-idf vectorizations of the dataset (compare with figure 2):

A screenshot of a cell phone

Description automatically generated

**Figure 4: 2-gram Vectorization**

**A screenshot of a computer

Description automatically generated**

**Figure 5: 3-gram Vectorization**

It was determined that there were not significant enough differences between n-gram vectorizations to explore further: all clustering was done on 1-gram TF-IDF vectorizations.

Visualization also considered different stemming types. Figures 6 and 7 show the 1-gram TF-IDF vectorizations with Lancaster and Porter stemming, respectively (compare with figure 2, which utilizes lemmatized data):

A screenshot of a computer

Description automatically generated

**Figure 6: Lancaster Stemmed Data**

A screenshot of a cell phone

Description automatically generated

**Figure 7: Porter Stemmed Data**

Again, there were not significant enough differences to warrant further exploration in the model creation stage.

The final visualization is that of document embeddings for both posts and subreddits. Figures 8 and 9 display document embeddings for subreddits and posts as reduced by singular value decomposition, respectively:

A picture containing sky, flying

Description automatically generated

**Figure 8: Subreddit Document Embeddings**

This vectorization exhibits some noticeable differences from the TF-IDF vectorization: it is more spread out and has considerably fewer overlapping points.

With these vectorizations, one can clearly see that there will be some problems in clustering all points accurately. For one, there are multiple overlapping clusters: in the TF-IDF vectorization, gaming, sports, and politics subreddits sit at the lower left corner, with a few points blending together; for the document embeddings, music and art subreddits are close to one another with points extending in different directions. As such, our models will not score very highly with silhouette score.

In addition to visualizing vectorizations, data exploration included creating word clouds. Word clouds allow one to view the most popular terms used in each subreddit, giving an intuitive understanding of how subreddits in the same category are related. It also provides support for the method chosen by showing the relationships within and between categories. Figure 9 shows a sample word cloud grouping for art subreddits:

A screenshot of a cell phone

Description automatically generated

**Figure 9: Word Clouds for Art Subreddits**

One can clearly see similar terms used in each art subreddit: words that express visual appeal (such as “look” and “beautiful”), words that express the process of making art (such as “make,” ‘draw,” and “paint”), and words that express positivity and approval (such as “love” and “like”). One can also see that “game,” from the r/pixelart subreddit, will make it cluster closely with gaming subreddits (pixel art itself is closely tied to the history of video games).

# Clustering

## K-Means

## Agglomerative Hierarchical

## DBSCAN & OPTICS

## Affinity Propagation

## Spectral Clustering

# Application

## Recommending Subreddits

## Recommending Posts

# Conclusion

# REFERENCES

1. A. Di Marco and R. Navigli, “Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction,” *Computational Linguistics*, vol 39, no. 3, August 6, 2013. [Online]. Available: <https://www.mitpressjournals.org/doi/10.1162/COLI_a_00148>. [Accessed September 30, 2019]
2. A. Huang, “Similarity Measures for Text Document Clustering,” *Proceedings of the 6th New Zealand Computer Science Research Student Conference*, January 1, 2008. [Online]. Available: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4633&rep=rep1&type=pdf>. [Accessed September 26, 2019]