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 [1].

## 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 [2].

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 [3].

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 [3]. 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 [3]. 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 [4]. 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 [5]. For lemmatization, NLTK’s WordNet lemmatizer was used [6]. 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. All word clouds were created using the implementation in [7]. 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 video games).

# Clustering

In order to produce recommendations, document clustering was performed. Both subreddits and individual posts were clustered, with differing results. Each model was implemented using SciKit Learn [8]. A total of five clustering models were trained; a complete discussion of each model is provided below. Table 1 provides an overview comparison chart of each model’s performance:

|  |  |  |
| --- | --- | --- |
| **Model** | **Subreddit Homogeneity** | **Posts Homogeneity** |
| K-Means | 0.87 | 0.54 |
| Agglomerative | 0.81 | 0.52 |
| DBSCAN | 0.11 | N/A |
| OPTICS | 0.35 | 0.20 |
| Affinity Propagation | 0.76 | 0.64 |
| Spectral Clustering | 0.85 | 0.52 |

**Table 1: Model Performance Overview**

## K-Means

The first clustering model fit to both subreddits and posts was a K-Means model with 8 clusters. For subreddits, this approach yielded homogeneity 0.69 and silhouette 0.40; for posts, homogeneity 0.54 and silhouette 0.31. Clearly, using 8 clusters does not yield a particularly effective clustering. Choosing 12 clusters yields 0.87 homogeneity and silhouette 0.40 for clustering subreddits; there is no significant change for using a different cluster number for posts. In terms of homogeneity, the K-Means model performed well on subreddit data but not on post data. It is also worth noting that post data incorporates significantly more data points and has a more complex, overlapping structure.

K-Means was also trained on TF-IDF vectorized data; due to the size of the TF-IDF matrix, it was impossible to use to train any other model. For subreddits, a K-Means model achieved homogeneity 0.85 and silhouette 0.53; for posts, homogeneity 0.60 and silhouette 0.37. The higher silhouette score for subreddits is interesting, as it would appear that the embedded models have greater separation between clusters; however, this is impossible to verify since plotting is done in two dimensions and higher dimensionality might hide differences that are not easily visualized.

## Agglomerative Hierarchical

An agglomerative clustering model was fit to both subreddit and post data. For both datasets, Ward clustering produced the best results. Again, a search for the ideal number of clusters was performed: eleven clusters was ideal for the subreddits model, while the number of clusters for posts was chosen to maximize v-score (since maximizing homogeneity would be equivalent to choosing the highest number of clusters possible).

For subreddits, the agglomerative hierarchical model achieved homogeneity 0.81 and silhouette 0.41; for posts, the model achieved homogeneity 0.52 and silhouette 0.25.

## DBSCAN & OPTICS

The density-based algorithms DBSCAN and OPTICS were fitted to both subreddit and post data. These models performed by far the worst of any models represented here. Again, it is difficult to visualize why this might be the case since higher dimensionality hides many aspects of the dataset.

To train the DBSCAN algorithm, a five-nearest neighbor analysis was performed to find the ideal value of epsilon. Choosing the value of epsilon that corresponded to the greatest change in distance did not produce ideal clustering; in fact, it resulted in one large cluster (which is clearly not useful for a recommendation system). As such, a slightly higher value of 0.6 was chosen for the subreddits model. The DBSCAN model achieved homogeneity 0.11 and silhouette score 0.086.

OPTICS is another density-based algorithm that does not require hyperparameter tuning. As such, it was ideal for use in the posts clustering (since its large size put time prohibitions on a neared-neighbor analysis for DBSCAN). For subreddits, OPTICS produced homogeneity 0.35 and silhouette 0.18; for posts, OPTICS produced homogeneity 0.20 and silhouette -0.47.

The low performance of both density-based algorithms precluded their use in recommendation.

## Affinity Propagation

Affinity propagation is a matrix-based algorithm that seeks to find examplars in the dataset and use them as centroids [9].

For subreddits, affinity propagation achieved homogeneity 0.76 and silhouette 0.48; for posts, affinity propagation achieved homogeneity 0.64 and silhouette 0.24.

It is worth noting that affinity propagation produced the highest homogeneity score for the posts dataset.

## Spectral Clustering

Spectral clustering implicitly reduces the dimensionality of inputs by finding their spectral decompositions; clustering is done in this lower dimension space [10].

The ideal number of clusters for each dataset was found: for subreddits, this was eleven clusters; for posts, this was whichever value maximized homogeneity during the search. Spectral clustering achieved homogeneity 0.85 and silhouette 0.41; for posts, homogeneity 0.52 and silhouette 0.31.

# Application

The original purpose of document clustering was to produce recommendations for both subreddits and posts. The best models (in terms of their homogeneity performance) were taken and used to create recommendations. For a given subreddit *s*, the recommended subreddits are all of the other subreddits in the cluster that contains *s*; the same method holds for a given post *p*. The models used for recommendations include:

1. K-Means
2. Agglomerative
3. Affinity Propagation
4. Spectral Clustering

## Recommending Subreddits

A subreddit was chosen at random to generate recommendations from. For example, consider r/pcgaming, which belongs to the gaming category. The recommended subreddits by model are presented below:

* K-Means:
  + gadgets
  + linux
* Agglomerative:
  + games
  + xboxone
  + gadgets
  + linux
* Affinity Propagation:
  + games
  + gaming
  + ps4
  + xboxone
  + gadgets
  + linux
  + technology
* Spectral Clustering:
  + gadgets
  + linux

There are a few important observations to be made. First, the cluster size produced by each clustering algorithm differs considerably: K-Means and Spectral Clustering both produced clusters of size two, while Affinity Propagation produced a cluster of size seven. Secondly, Agglomerative and Affinity Propagation both produced clusters that included gaming subreddits, while K-Means and Spectral Clustering did not. It is interesting that each clustering also includes r/linux and r/gadgets, perhaps reflecting the nature of r/pcgaming as a computer-hardware centered subreddit.

The K-Means model trained on the TF-IDF Vectorization was also used to present recommendations for a given subreddit. This time, the randomly chosen subreddit was r/cfb in the sports category. The recommended subreddits were:

* nba
* nfl
* soccer

This makes sense, as all are also sporting subreddits.

## Recommending Posts

A post was chosen at random and used to generate recommendations. The volume of posts (over 3700) makes it difficult to assess the usefulness of these recommendations, as each cluster has hundreds of posts. Nonetheless, one can get an overall sense of how relevant a set of recommended posts is by looking at the category of the posts. For example, a randomly chosen post is from the r/psychology subreddit (in the science category). The list of recommended posts from the K-Means model is huge, but includes posts from:

* games
* gaming
* pcgaming
* ps4
* xboxone
* news
* PoliticalHumor
* worldnews
* biology
* chemistry
* physics
* psychology
* science
* futurology
* gadgets
* tech
* technology

This range is huge, and many of the topics presented do not seem intuitively related to psychology. However, without the titles of these posts it is impossible to form an accurate opinion of their relevance. It may be that many of the gaming posts have language that is related to psychology (such as posts expressing frustration, overcoming difficulty, anger at a company or practice, or users talking about habits and routines). It is also important to note that each post relates to a specific topic or discusses a specific event, making it difficult to relate to even those posts that come from the same subreddit (for example, two posts in r/psychology might be discussing two completely unrelated studies in psychology).

Cluster size remains huge across all models (including the K-Means model trained on TF-IDF Vectorized documents), as does the mixed-category clustering.

# Conclusion

Document clustering is a versatile technique that allows for the exploration of textual data to discover patterns and relationships. As such, it is a valuable data mining technique that has a number of interesting and useful applications. In this project, it was applied to textual data from a content aggregation site to find patterns amongst online communities.

Document clustering also involves the interaction of many robust, deep, and integral parts to form an analysis pipeline. The steps it utilizes, such as data scraping and cleaning, linguistic analysis to form lemmas and stems, vectorization of textual data into numerical representations, and application of clustering models, come from areas as diverse as website design, computer science, linguistics, and statistics. A working understanding of each pipeline component is necessary to create accurate representations of documents and discover meaningful information from textual datasets.

While clustering models generally performed well in terms of homogeneity, they all fell short in the internal measure of silhouette score. Low silhouette scores can generally be regraded as a reflection of overlapping clusters in the dataset. This is most likely due the limited size of the collected dataset. To elicit the differences between subreddits, there needs to be a wider range of posts collected; this would ensure that each subreddit has both top-rated posts (which often reflect big news events) and average posts (which often reflect the mundane concerns of a community). With this in mind, the cost of scraping Reddit using PRAW is quite expensive: API requests are limited, there are a number of difficulties in getting descriptive information from each post (such as missing or deleted usernames, missing post IDs, etc.), and text cleaning becomes increasingly expensive as the total volume of text increases. Additionally, not every post on a subreddit can count towards its total text data: some posts have none or very few comments, and there are hundreds of reposts on popular or controversial news events. Finally, the question of which subreddits to scrape becomes difficult. Even within hand-assigned categories, there are significant differences that make topic coherence difficult to achieve.

The training of document embeddings also requires greater attention. This project treated the dimensionality of the Doc2Vec algorithm as a hyperparameter to maximize a K-Means model’s homogeneity. This is obviously a flawed approach, since it heavily favors the K-Means model (which itself favors globular distributions, meaning that dimension was probably chosen to create the vectorization that produced the most globular clusters). TF-IDF vectorization was limited to a K-Means model, since the total size of the term-frequency matrix precluded the use of any other model. In order to more fully utilize this approach, a better understanding of how to efficiently perform dimensionality reduction is needed.

This project produced a rudimentary application: a recommendation system for individual subreddits and posts. Subreddit recommendation was intuitive while post recommendation failed to offer useful results. It is worth noting that content recommendation itself is a robust field that generally incorporates more descriptive and individual data than textual information alone can offer. For example, collaborative filtering utilizes implicit and explicit feedback from a service’s users to help produce recommendations; no attempt to use user data was made in this project. Additionally, the data collection step did not collect specific information about each post that might have aided in post recommendations; most notably, the titles of each post were not collected. Having titles would have helped in post recommendations by giving a human-understandable description of each post’s topic.

# REFERENCES

1. Boe, Bryce, “PRAW Documentation,” December 6, 2019. [Online]. Available: <https://praw.readthedocs.io/en/latest/getting_started/quick_start.html>. [Accessed December 8, 2019]
2. 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]
3. 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]
4. L. Quoc and T. Mikolov, “Distributed Representations of Sentences and Documents,” Google Inc., May 22, 2014. [Online]. Available: <https://cs.stanford.edu/~quocle/paragraph_vector.pdf>. [Accessed December 8, 2019]
5. “NLTK Stem Package,” NLTK. [Online]. Available: <https://www.nltk.org/api/nltk.stem.html>. [Accessed December 8, 2019]
6. “NTLK Wordnet Package,” NTLK. [Online]. Available: <https://www.nltk.org/_modules/nltk/stem/wordnet.html>. [Accessed December 8, 2019]
7. A. Mueller, “Wordcloud Documentation”, November 2019. [Online]. Available: <http://amueller.github.io/word_cloud/>. [Accessed December 8, 2019]
8. “SciKit Learn: Clustering”. [Online]. Available: <https://scikit-learn.org/stable/modules/clustering.html>. [Accessed December 8, 2019]
9. F. Brendan and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science*, vol 315, February 16, 2007. [Online]. Available: <http://utstat.toronto.edu/reid/sta414/frey-affinity.pdf>. [Accessed December 8, 2019]
10. U. von Luxburg, “A Tutorial on Spectral Clustering,” *Statistics and Computing*, vol 17, no. 4, 2007. [Online]. Available: <http://people.csail.mit.edu/dsontag/courses/ml14/notes/Luxburg07_tutorial_spectral_clustering.pdf>. [Accessed December 8, 2019]