Machine Learning Engineer Nanodegree
Capstone Project Report

Lee Burke

November 29th, 2017

1 Definition

1.1 Project Overview

Natural language processing is an essential but difficult application of machine learning. Language is as complex as the meaning which it conveys; as our understanding of the world is contingent on incomplete and shifting knowledge, so too is our language a fuzzy shadow of crisp logical formalism. Indeed, teaching a machine to fully parse language is perhaps equivalent to teaching it to comprehend, like a child learning to speak. Clearly, the automatic extraction of semantic features from raw text is difficult.

Nonetheless, much progress has been made. In 1963, Mosteller and Wallace used Bayesian statistics to analyze the authorship of *The Federalist Papers*[9]. Decades later, these foundations are still recognizable (e.g., the hierarchical Bayesian approach of Latent Dirichlet Allocation (LDA) in [3]). The data revolution (see, e.g., [7]) has leveraged these statistical techniques to power new technologies: everyone has a virtual assistant in their pocket capable of responding to commands.

This project explores modern techniques for language processing, including `word2vec`, `doc2vec` and related technologies, and older tools like LDA. It brings these tools to bear on the automatic classification of text in the same vein as Mosteller and Wallace's research decades ago. More recent similar projects include attempts to automatically categorize Hacker News articles[1] and a playful new book exploring statistical insights into literature.[2]

In particular, the semantic qualities of various Reddit.com communities are compared. Reddit is one of the most-trafficked domains on the internet, with thousands of interconnected message boards and millions of submissions per day[4]. Analysis is made interesting by the organically user-generated structure of the site, the semi-anonymous nature of Reddit user accounts, and the free-wheeling nature of casual conversation. Surfing Reddit is like walking through the biggest crowd on Earth, listening to everyone else's conversations. This project attempts to make navigating the crowd more convenient.

Reddit is organized like a forest graph: anyone can found a *subreddit*, the root of a tree. To this subreddit, anyone can submit a *post*, the first level of branches. These posts can be blocks of text or links to other websites (often images or videos). On each post, anyone can leave a *comment*, and anyone can comment on other comments. The conversations happening on Reddit occur almost exclusively in this comment section: submissions are usually short, and have much less back-and-forth among users. Subreddits often develop their own sub-cultures (e.g., /r/catsstandingup, where submissions must be pictures of cats standing up on two legs, and the comment section is edited to say only, "cat.", hundreds of times). These sub-cultures form the primary application of this project.

[1] `https://techcrunch.com/2017/05/14/building-a-smarter-hacker-news/`

[2] Ben Blatt. *Nabokov's Favorite Word is Mauve*. New York, New York: Simon & Schuster, 2017.

1.2 Problem Statement

There is no built-in Reddit utility to discover new subreddits. New communities are usually discovered organically, through other users' mentions, or through keyword search. Instead, this project clusters subreddits based on their comment sections in an effort to find communities similar to the comment section of a given post or subreddit. This is an unsupervised clustering problem that could fit into a larger recommender system based on user activity, similar to a "suggested for you" dialogue on other websites.

Mitchell defines a machine to be learning if its performance on a *task* (evaluated by some *measure*) improves with *experience*, as cited in [6]. Here, we cluster subreddits (the task), learning only from the text of the comment sections of those subreddits (the experience), and assessing this clustering according to a similarity score (the performance measure, see Section 1.3). In the process of feature engineering, we also explore classifying subreddits (using a usual classification scorer like accuracy) using the text of the comment sections to predict their subreddit labels.

1.3 Metrics

Different metrics are utilized to evaluate feature engineering methods and clustering methods. Other metrics offer different insights into the solution.

1.3.1 Feature Engineering Evaluation

Due to the inherently subjective nature of unsupervised learning problems (if we had an objective, it would no longer be unsupervised!), this project compares feature engineering methods by framing them as a supervised learning problem, predicting from which subreddit a given comment was taken. Only after suitable features have been found do we apply clustering algorithms. We do this because supervised evaluation metrics are much easier to come by than metrics for clustering: accuracy is defined to be simply the number of correct predictions divided by the total number of predictions.

It is often interesting to compare F-scores among algorithms, which more clearly show the tradeoff between precision and recall. The usual F1 score is defined as $2PR/(P+R)$ where P is precision and R is recall. For the multi-class case, we may simply take the average of the score for each class, weighted by the size of the class. However, precision is undefined (and so too the F-score) if no positive identification is made. In a classification problem with many class labels and low accuracy, this is a likely scenario. Thus, we use only accuracy to compare methods.

1.3.2 Internal Clustering Evaluation

Ideally, we would *indirectly* measure the utility of our solution in practice; we could ask users how similar the suggested subreddits seem to them. This is infeasible for early stages of product development. We could *manually* assess the quality of clustering, but without enough human supervision (at which point, why not indirectly evaluate through actual product release?) this is subject to strong bias and noise. Once these studies have been done, there are a number of *external* evaluation techniques to compare computed clusters to ground-truth labels.

For fully automated, fully unsupervised learning, it is not clear how best to compare clustering methods *internally*, because any measure of clustering goodness could (theoretically) be used as a clustering objective itself. Considering clustering methods as optimization problems, we are, in effect, comparing how similar each method's objective function is to the chosen evaluation function [5]. Thus, much care should be taken when interpreting internal evaluations of clustering algorithms.

Nonetheless, we can discuss what good clustering looks like. In the most abstract case, we consider each data point to be an n-dimensional vector in some normed vector space (where each dimension represents a feature and the norm gives us a notion of distance). To cluster these points best, we want cohesion within and separation between clusters; that is, we want small intra-cluster distances and large inter-cluster distances. An easy to understand, worst-case evaluation metric is the Dunn index, where we take the ratio of the worst-case separation to the worst case cohesion: Let δ_{ij} be any measure of the distance between points in

clusters i and j, and let Δ_k be any measure of the distance between points within cluster k. Then the Dunn index D is defined as

$$D = \frac{\min_{i \neq j} \delta_{ij}}{\max_k \Delta_k}.$$

Better methods receive a larger Dunn index, as clusters become more tightly clustered and better separated. There are many other metrics, including the Davies-Bouldin index, an average of worst cases per cluster, and the Calinski-Harabaz index, which uses inter- and intra-cluster dispersion matrices.

This project instead uses the silhouette score, a normalized ratio of cohesion and separation for each point: given data point x_i, find the average distance to other points in its own cluster a_i, and the average distance to other clusters B_{ij}. Define $b_i = \min_j B_{ij}$ (the average distance to the nearest cluster). Then the silhouette score is given by

$$S_i = \frac{b_i - a_i}{\max\{a_i, b_i\}}.$$

This score is bounded within $[-1, 1]$, with higher scores indicating good separation and cohesion. Aggregate scores are found by averaging the point-wise scores. This metric is easily interpretable because poorly clustered points are already labeled as such. As discussed above, this metric privileges some clustering algorithms (e.g. K-Means) over others (e.g., density-based methods).

1.3.3 Other Evaluations

This project also considers the cost of each method: when processing the massive data available from a site like Reddit, slow algorithms may be prohibitively expensive to implement. Cost will be measured as clock time required for both feature engineering and the actual clustering.

Qualitative analysis may also be interesting, so visualization of word/document embeddings are explored, as well as interpretability thereof: can we say with confidence why a given comment or subreddit is classified a certain way?

2 Analysis

2.1 Data Exploration

Reddit provides an API for accessing its data, and this is the best way to download small amounts of real-time data. However, Reddit enforces a 30 request per minute limit, so trolling through millions of posts may be difficult. To avoid this rate limiting, user `Stuck_in_the_Matrix` has made available massive monthly data dumps of Reddit comments on the Google cloud [12]. This source will provide the corpus for training and testing the model.

In October 2017, there were about 86,000,000 comments posted to Reddit, totaling around 23 GB of data. To pare this dataset down to be under 500 MB, we use the Google BigQuery platform to filter on time, subreddit, and score.[3] The exact SQL query used to extract the data is shown in Listing 1. It selects comments on three slices: time (October 2017), subreddit (the top 1000 subreddits, ranked by number of comments) and score (the top 2000 comments in each subreddit, ranked by score), for a total of about two million examples. To stay within memory constraints on BigQuery, these are taken from a 10% random sample of all the comments posted to Reddit in October 2017. This random sampling leads to some variation in the number of comments per subreddit, as shown in Figure 1. We then export the data from BigQuery and load into a `Pandas` dataframe.

After this ad-hoc ETL process transforms the big data dump into more manageable small data, the dataset consists of about 1.9 million data points with only two features: a variable-length block of text and a subreddit label. Other information (time stamps, usernames, and scores) have been stripped for simplicity.

[3] In addition to posting submissions or comments, users may vote either up or down on any given post. The difference between the numbers of upvotes and downvotes is called the score.

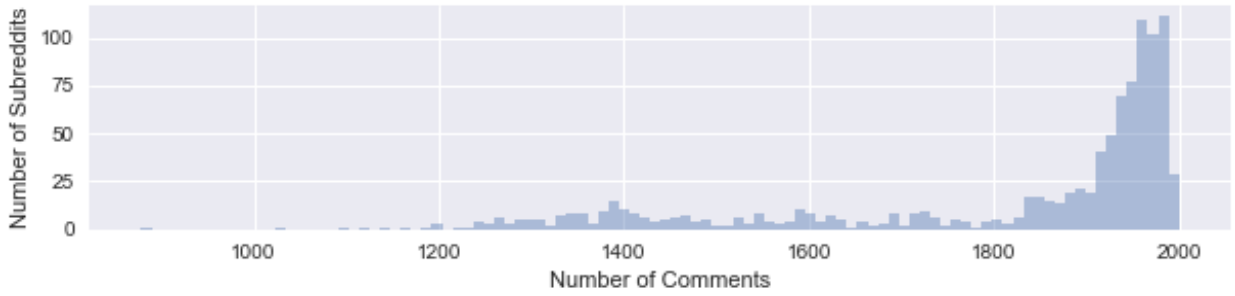

Figure 1: The distribution of comments over subreddit label is shown for this project's dataset, after preprocessing.

2.2 Exploratory Visualization

To justify the conceit of clustering subreddits based on the contents of their comment sections, we present an initial analysis of some particularly cohesive subreddits and how they compare to Reddit as a whole. Using Andreas Mueller's wordcloud package [10], we present the most frequent words in each collection of comments as a wordcloud.

Figure 2 shows the most frequent words overall (except the stopwords built into the wordcloud package). Figures 3 and 4 show the most frequent words in the `gaming` and `politics` subreddits, respectively. We

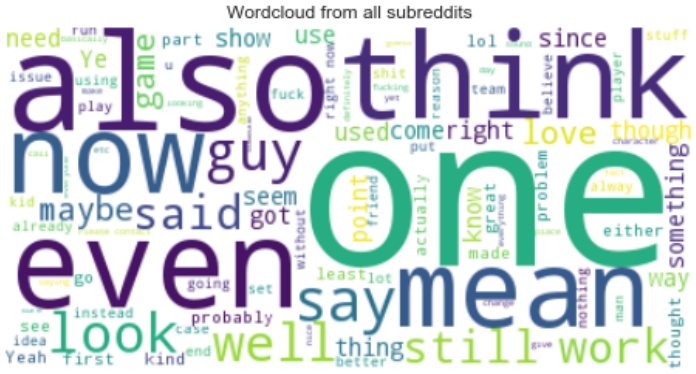

Figure 2: The most frequent words in the entire dataset are shown with height proportional to frequency.

see already that the word "game" may help distinguish `gaming`, and that "Trump", "Republican", and "president" may distinguish `politics`.

2.3 Algorithms and Techniques

As discussed above, this project proceeds in two phases: feature extraction (measured using visualization sanity checks and classification scores) and clustering (measured with a similarity score).

2.3.1 Methods for Feature Extraction

The simplest technique for feature extraction is a bag-of-words term frequency matrix (tf): the frequency that each word appears in each document is stored in a matrix (words vs. documents). Each document is

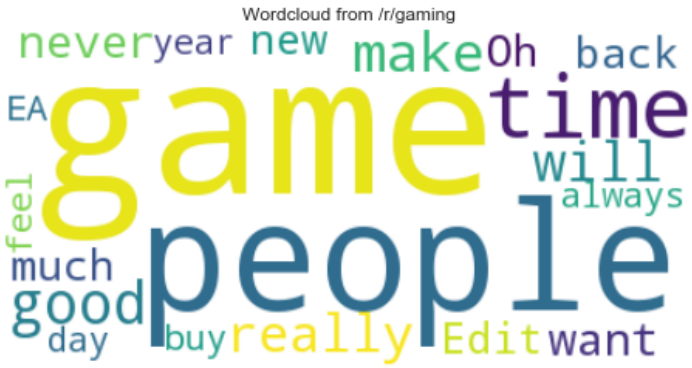

Figure 3: The most frequent words in /r/gaming are shown with height proportional to frequency.

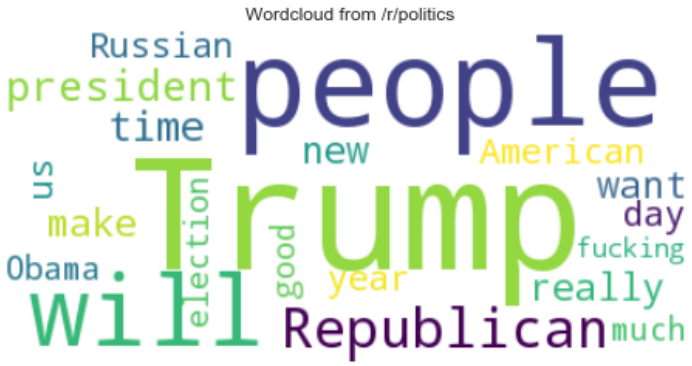

Figure 4: The most frequent words in /r/politics are shown with height proportional to frequency.

then represented by a vector of word frequencies. This is sometimes referred to as a co-occurrence matrix, showing the probability of a word and document occurring together. Word order and other syntactical information is completely lost in this representation. A common improvement to this representation is term frequency/inverse document frequency (tf/idf), which adds weight to the frequency of a word in a document according to how infrequently it appears in the entire corpus; this puts less emphasis on words common in all documents.

The tf/idf representation is still unwieldy because it is so high-dimensional (if often quite sparse). We seek a map from the word/document vector space to some lower dimensional representation which retains as much useful information as possible from the original co-occurrence matrix.

First, Latent Semantic Analysis (LSA, sometimes called Latent Semantic Indexing, as it was pioneered for information retrieval)[5] approaches this problem using a truncated singular vector decomposition (or SVD, a common linear algebra tool related to an eigen- decomposition; the details of computing such a decomposition are not discussed here).

A second technique is probabilistic LSA (pLSA)[5], which models a given word/document co-occurrence as a mixture of conditionally independent multinomial distributions:

$$P(w,d) = \sum_c P(c)P(d|c)P(w|c) = P(d)\sum_c P(c|d)P(w|c) \tag{1}$$

Interestingly, modeling the co-occurrences in this way is equivalent to another standard decomposition from

linear algebra: the non-negative matrix factorization using a generalized Kullback-Leibler divergence (NMF-KL). Showing why these are equivalent and the details in computing the decomposition are beyond the scope of this project.

The third technique is Latent Dirichlet Analysis (LDA)[3], pLSA with a Dirichlet prior; which is to say that the probability that each topic is found in a document is given by a (sparse) Dirichlet distribution, and the same for each word in a topic. In effect, this assumes that only a few words make up each topic, and only a few topics each document. The LDA model is equivalent to certain tensor decompositions[4], but the LDA is usually accomplished using an expectation-maximization algorithm similar to those found in certain deep learning models.

The above algorithms are implemented in scikit-learn[11].

Finally, neural network-based models have emerged as the state-of-the-art in many fields of machine learning, and NLP is not immune. Google's `word2vec`, and associated `doc2vec`, (among many other architectures) leverage neural structures for efficient, accurate vector representations of documents and words[8]. In particular, word2vec assigns a vector to each word in a document, then trains these to predict nearby words using a dense, three-layer network. These vectors then work as proxies for the meaning of the words, so that "Queen minus Woman plus Man equals King". The doc2vec framework simply keeps an extra vector to track the semantic content of the document as well. These frameworks are visualized in Figure 9. The implementation of doc2vec used in this project is from gensim[13].

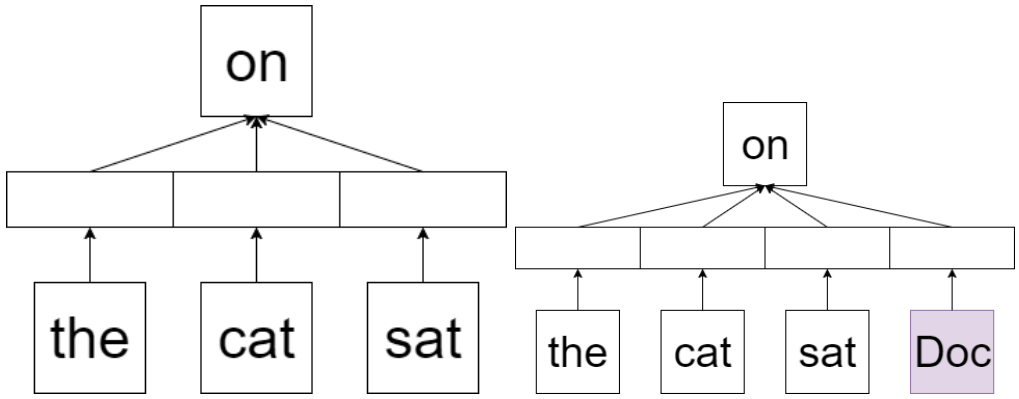

Figure 5: Frameworks for training word2vec (left) and doc2vec (right) are reproduced from [8].

The central challenge of this project is to compare the speed and efficacy of these several methods, as it is difficult to know a priori what method will work best for this particular application.

2.3.2 Methods for Clustering

As discussed in Section 1.3.2, ranking clustering methods is difficult, even after implementation. To do so a priori is clearly a fallacy. Instead, this project simply explores two standard options for online clustering, listed below.

The pre-eminent baseline clustering method is K-means, which attempts to form a specified number (K) of clusters by iteratively assigning a centroid (the means) to each sample that minimizing a target function to improve cohesion and separation. Another method is Birch, which builds a hierarchical tree of sub-clusters, then applies an agglomerative clustering algorithm to build the final clusters. Both of these are implemented in scikit-learn[11]. Most other clustering methods are severely limited by available memory: these two have variants which can be trained in mini-batches, using only a portion of the data at any given time.

Spotify's `Annoy` is an approximate nearest neighbor solver which allows for a memory-efficient, fast indexing of nearby vectors[1]. This is used for a final recommendation process: we simply find the nearest n

[4]See AWS SageMaker, `https://docs.aws.amazon.com/sagemaker/latest/dg/lda-how-it-works.html`

vectors to a given subreddit's location in the feature space.

2.4 Benchmark

2.4.1 Feature Extraction Benchmark

The simplest model for the supervised problem (predicting from which subreddit a given comment was taken) takes word frequencies, then finds a linear discriminant or applies Naive Bayes to classify each document (each of these are used in the famous Mosteller and Wallace paper[9]). A simple implementation of this method with multinomial Naive Bayes is shown in Listing 2, where `X` contains comments and `y` is a label encoding of the subreddit names.

2.4.2 Clustering Benchmark

As discussed in Section 6.2, comparing clustering algorithms is difficult, but we may nonetheless use the commonplace K-Means algorithm as a simple model to compare against. The most straightforward feature extraction method, LSA, is performed (i.e., we attempt to cluster rowsof the truncated SVD of the tf/idf matrix).

3 Methodology

3.1 Data Preprocessing

The data for this project is raw text. Before analysis may proceed, we must filter out Reddit markdown language[5], non-English characters, and the trace left behind when one deletes a comment. We find that two subreddits, /r/newsokur and /r/Womad, are written almost entirely in Korean, so for simplicity we filter them out. Furthermore, 3.85% of the comments were deleted or removed by moderators, so these are removed from the dataset[6]

Then comes the non-trivial task of parsing the remaining text into term frequency vectors. There are many approaches to this process of *vectorizing*, with few obvious advantages one way or another. Here, we use the scikit-learn vectorizers using the built-in English stopwords set (common words not to be included), ignoring words that appear in over half the documents or in only one. Instead of the built-in tokenizer (which actually separates words), we use the WordNet lemmatizer from the Natural Language Toolkit[2] to avoid counting alternate conjugations as separate words as much as possible, then split on whitespace.

The output of this preprocessing is a tf/idf matrix, where each column represents a word and each row a document. The gensim doc2vec implementation instead takes the data as a list of gensim labelled sentences, a gensim-specific datatype which is basically a list of tokenized words. A gensim utility is used to form this list.

3.2 Implementation

Thanks to the pre-built interfaces of scikit-learn, no significant challenge was posed in implementation of most of the methods. To match this interface, I built a wrapper class for the gensim doc2vec utilities, which includes the requisite `init`, `fit`, and `transform` methods, and holds the gensim doc2vec model as data. To avoid clunky parameter passing, I exposed only the number of iterations, the size of the feature vectors, and the model flavor (cBoW or skip-gram, see [8]) as parameters. Having unified interfaces allowed me to use scikit-learn's parameter search utility `RandomizedSearchCV` for model refinement.

[5]See `https://www.reddit.com/r/reddit.com/comments/6ewgt/reddit_markdown_primer_or_how_do_you_do_all_that/c03nik6/`

[6]To preserve the hierarchical structure of Reddit, deleted comments are replaced with the text "[deleted]".

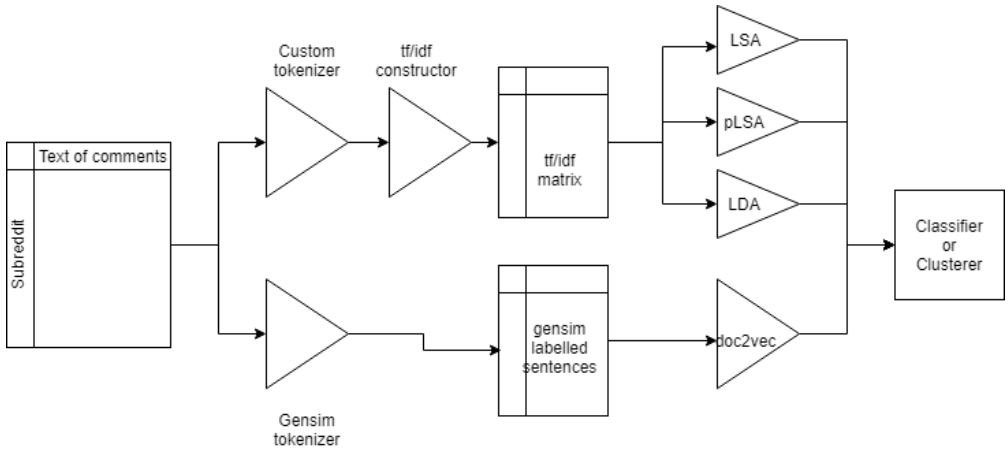

Figure 6: The flow of data through the implementation is shown.

3.3 Refinement

Rather than pursue an initial/intermediate/final improvement paradigm, the models are refined by a search of hyperparameter space, then a selection from among the various model options. This is an expensive process. The cost of the final production model must be multiplied by the number of cross-validation splits and the number of parameter combinations, which grows exponentially with the number of parameters. to combat this cost, the data is down-sampled by a factor of 100, only a few hyperparameters are changed from their default values, and only some of their possible values are evaluated.

Nonetheless, relevant results are obtained. Three classification experiments are executed: one to explore LSA and K-Nearest Neighbors (Figure 7), another to explore LDA (Figure 8), and a third to explore doc2vec (Figure 9).

From the first experiment (Figure 7, we see that the cost of KNN is not significantly dependent on the number of neighbors chosen, but that up to a value of 4 or 5, increasing this value leads to higher accuracy with LSA. Furthermore, we see a linear increase in the cost of LSA as the number of components is increased, but the gain in accuracy diminshes. We choose a middle ground of 60 for future experiments.

The second experiment (Figure 8) explores teh learning decay and learning offset of the LDA model. The cost of running so many LDA trainings is prohibited, and significant improvements in accuracy were not obtained, beyond the general guidance to set both parameters as low as possible.

The third experiment (Figure 9) explores the number of iterations in training the neural net, and the dimensionality of the document vector. This experiment is also constrained by the high cost of training the neural net, with even less clear-cut conclusions to be drawn. However, we note that too many iterations or too long of a vector may be overfitting the model; we set the parameters to 60 and 75, respectively.

Experiments such as these are much less useful in clustering problems, as by definition no objective metric is available to compare. However, we use the silhouette score to compare among the four feature extraction methods (LSA, pLSA, LDA, and doc2vec), the two clustering methods (KMeans and Birch), and for various average sizes of clusters.

In the KMeans experiment (Figure 10), we see LSA, pLSA, and do2vec exhibiting similar performance with a peak cluster size around 20 (pLSA appears to perform somewhat better, and doc2vec seems less affected by cluster size). In general, LDA is outperformed by the other methods in silhouette score.

Similarly, the Birch experiment (Figure 11) shows the same results for LSA, pLSA, and doc2vec (though the peak around a cluster size of 20 is even less apparent for doc2vec). Now see LDA performance appear to skyrocket, but this is evidence of the poor suitability of the silhouette score for comparing performance: the clusters quickly conglomerate into very few, very large clusters, with many singletons. This leads to a high score, but is not very useful for exploring Reddit.

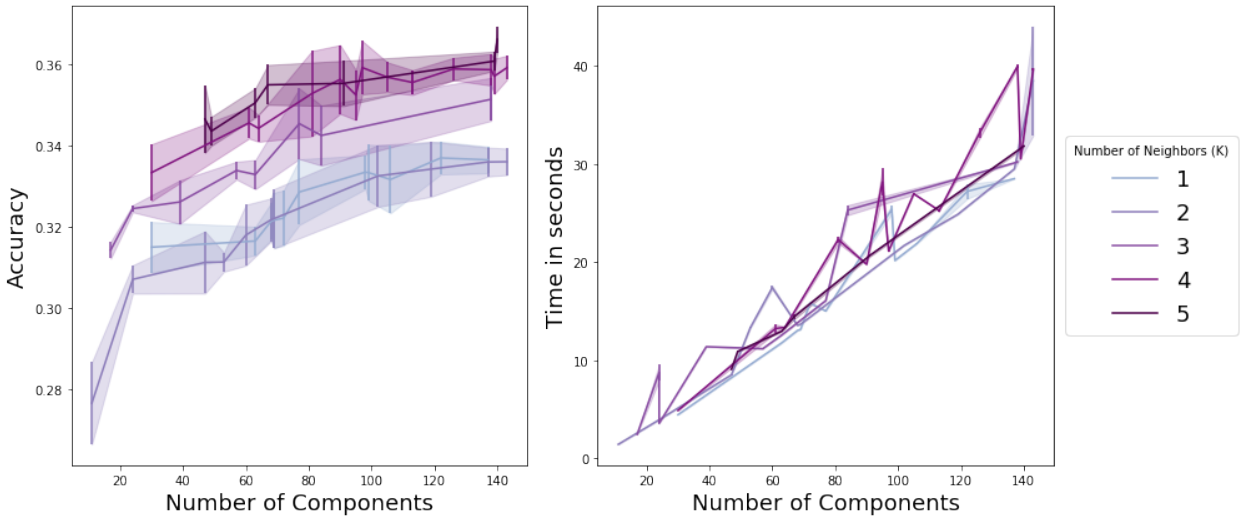

Figure 7: Accuracy and time are plotted against the number of LSA components for various K in KNN. Shaded regions represent one standard deviation in cross-validation.

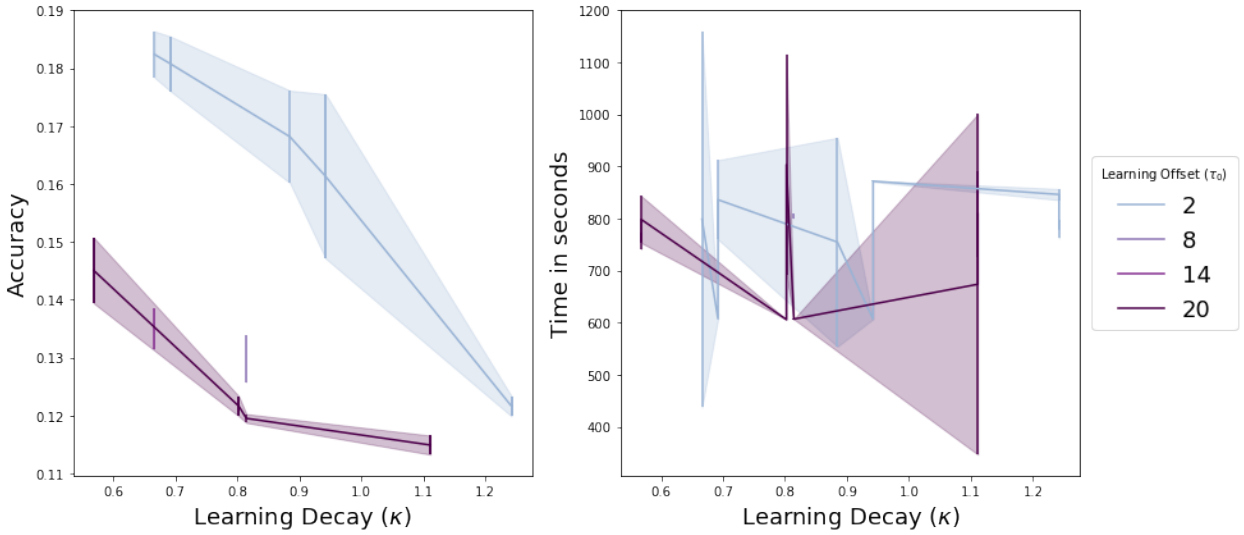

Figure 8: Accuracy and time are plotted against the decay rate (κ) for various learning offsets (τ_0). Shaded regions represent one standard deviation in cross-validation.

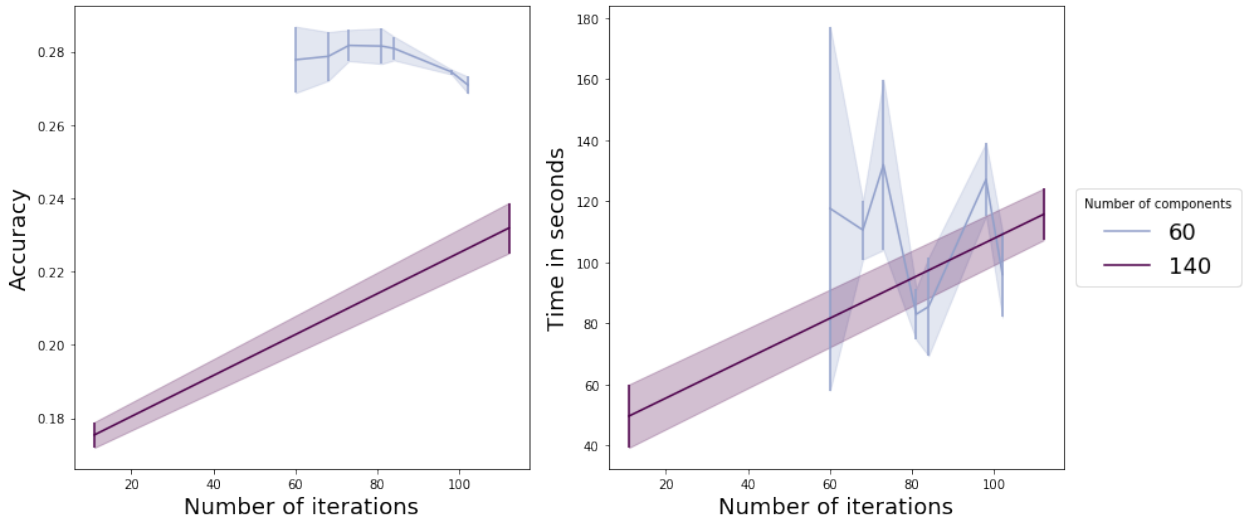

Figure 9: Accuracy and time are plotted against the number of training iterations for various numbers of components. Shaded regions represent one standard deviation in cross-validation.

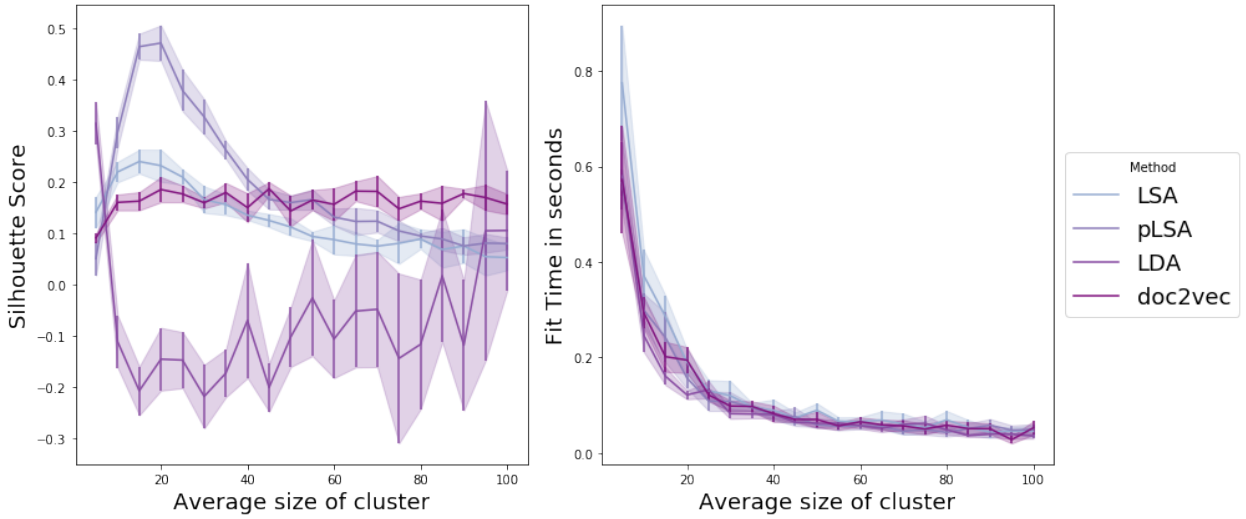

Figure 10: Silhouette score and time are plotted against the average size of each cluster for various feature extraction methods, using online K-Means. Shaded regions represent one standard deviation in cross validation.

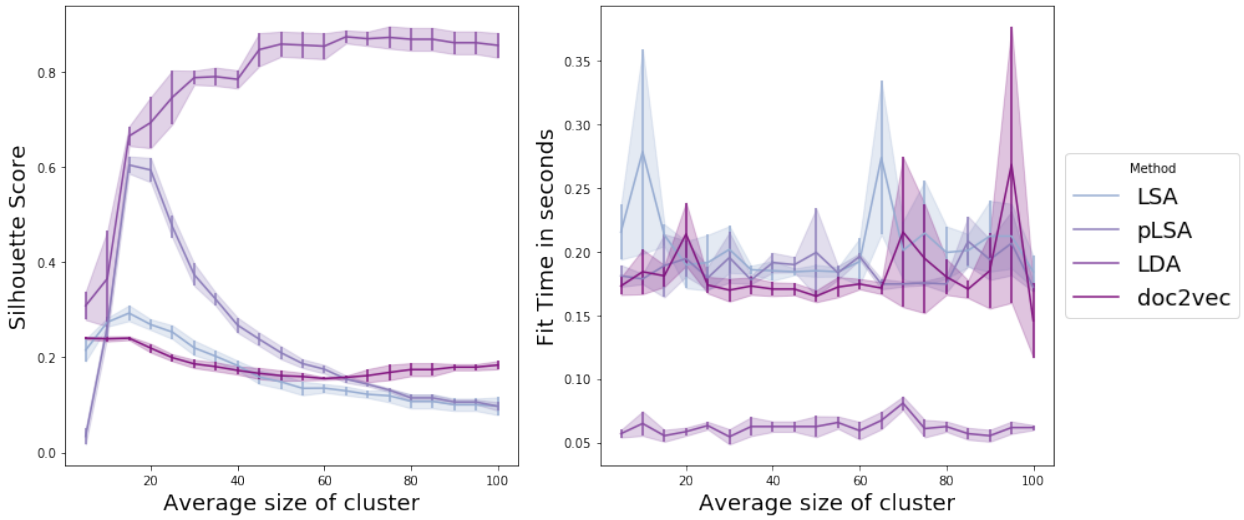

Figure 11: Silhouette score and time are plotted against the average size of each cluster for various feature extraction methods, using Birch. Shaded regions represent one standard deviation in cross validation.

In summary, using the tuned hypeparameters detailed above, LSA with 60 components achieves an accuracy just under 30% in a 998-choice classification task. This is comparable to tuned pLSA and doc2vec models and much better than LDA, at a fraction of the cost of any of these. Qualitative exploration of KMeans and Birch are necessary to determine relative quality between these, as no meaningful difference is found using quantitative comparisons of silhouette score. We use Birch for final clustering, as its speed does not depend on the number of clusters.

4 Results

4.1 Model Evaluation and Validation

We now turn to qualitative evaluation of the final clustering of Reddit communities, using LSA for feature extraction and Birch for clustering. Note that the refinements of Section 3.3 used five-fold cross-validation; that is, various subsets of the total data are used as training data with the held out portion used to measure accuracy. The shaded regions in the plots from that section show the standard deviation of results, and as such measure robustness to changes in the training data. This is not explored here, as qualitative exploration does not lend itself easily to such testing.

We examine three clusters to examine the efficacy of analysis. First, a cluster of communities focused around discussion of cryptocurrencies (Figure 12). The cluster contains all of the cryptocurrency-related subreddits from the data set, as best as we can tell. This is a huge success: a user interested in Bitcoin could easily find information on related topics.

Next, two clusters highlight the success of clustering: not only are music-related clusters created, we find both communities which talk about listening to music (Figure 13) and communities which talk about making music (Figure 14). It is clear that some degree of granularity is possible.

Finally, we can use a nearest neighbors index (here we use the *Annoy* approximate nearest neighbors package[1]) to directly find most similar subreddits. Starting from the popular exercise community /r/Fitness, we find communities focused on women's fitness and various flavors of weightlifting and gym-going (Table 1).

‘vertcoin’,
‘ethereum’
‘CryptoCurrency’
‘Bitcoin’
‘ethtrader’
‘Monero’
‘waltonchain’
‘BitcoinMarkets’
‘Iota’
‘BitcoinAll’
‘Ripple’
‘NEO’
‘btc’

Figure 12: The cryptocurrency cluster: subreddits, and most common words.

’Kanye’
’TaylorSwift’
’FrankOcean’
’indieheads’
’hiphopheads’
’bangtan’
’Metalcore’
’listentothis’
’Eminem’
’Metal’
’popheads’
’radiohead’
’deathgrips’
’kpop’
’Music’

Figure 13: Music listener cluster: subreddits, and most common words.

Table 1: Approximate cosine distance between /r/Fitness and its nearest neighbors

Subreddit	Distance
/r/xxfitness	0.21
/r/bodybuilding	0.22
/r/weightroom	0.36
/r/orangetheory	0.39
/r/powerlifting	0.47

'synthesizers'
'makinghiphop'
'WeAreTheMusicMakers'
'brandnew'
'edmproduction'
'Guitar'
'vinyl'
'guitarpedals'
'headphones'

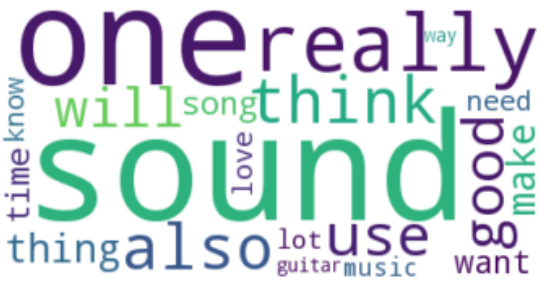

Figure 14: Music maker cluster: subreddits, and most common words.

Subreddit	Distance
/r/xxfitness	0.21
/r/bodybuilding	0.22
/r/weightroom	0.36
/r/orangetheory	0.39
/r/powerlifting	0.47

Figure 15:

4.2 Justification

5 Conclusion

5.1 Free-Form Visualization

5.2 Reflection

5.3 Improvement

Listings

```
SELECT
  body,
  subreddit,
FROM (
  SELECT
    body,
    subreddit,
    score,
    ROW_NUMBER() OVER(PARTITION BY subreddit ORDER BY score DESC) rows_score
  FROM
    [fh-bigquery:reddit_comments.2015_05]
  WHERE
    subreddit IN (
    SELECT
      subreddit
    FROM (
      SELECT
        COUNT(*) num_coms,
        subreddit
      FROM
        [fh-bigquery:reddit_comments.2015_05]
      GROUP BY
        subreddit
      ORDER BY
        num_coms DESC
      LIMIT
        100)))
WHERE
  rows_score <= 20000
```

Listing 1: BigQuery Example (44.1s elapsed, 9.94 GB processed)

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_validate

vectorizer = CountVectorizer()
clf = MultinomialNB()
counts_multinomialNB = Pipeline([('counts', vectorizer), ('MultiNB', clf)])

scores = cross_validate(counts_multinomialNB, X, y)
```

Listing 2: Feature Extraction Benchmark model.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

X = data.body
vectorizer = CountVectorizer()
X_freqs = vectorizer.fit_transform(X)

svd = TruncatedSVD(1000)
X_svd = svd.fit_transform(X_freqs)

clusterer = KMeans(n_clusters=10, random_state=0)
clusterer.fit(X_svd)
preds = clusterer.predict(X_svd)

score = silhouette_score(X_svd, preds)
```

Listing 3: Clustering Benchmark model.

References

[1] Erik Bernhardsson. *Annoy*. https://pypi.python.org/pypi/annoy. [Online; accessed 24-November-2017].

[2] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[3] David M Blei et al. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022. ISSN: 15324435. DOI: 10.1162/jmlr.2003.3.4-5.993. arXiv: 1111.6189v1.

[4] Upvoted: The Official Reddit Blog. *Reddit in 2015*. https://redditblog.com/2015/12/31/reddit-in-2015/. [Online; accessed 24-November-2017]. 2017.

[5] Ronen Feldman and James Sanger. *Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. New York, NY, USA: Cambridge University Press, 2006. ISBN: 0521836573, 9780521836579.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Tony Hey, Stewart Tansley, and Kristin Tolle, eds. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Redmond, Washington: Microsoft Research, 2009. URL: http://research.microsoft.com/en-us/collaboration/fourthparadigm/.

[8] Quoc V. Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[9] Frederick Mosteller and David L. Wallace. "Inference in an Authorship Problem". In: 58.302 (1963), pp. 275–309. ISSN: 1537274X. DOI: 10.1080/01621459.1963.10500849.

[10] Andreas Mueller. *word cloud*. https://github.com/amueller/word_cloud. 2018.

[11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[12] *Reddit Comments Dataset*. https://bigquery.cloud.google.com/table/fh-bigquery:reddit_comments.2017_10. 2017.

[13] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50.