Machine Learning Engineer Nanodegree
Capstone Proposal

Lee Burke

November 29th, 2017

1 Domain Background

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*[8]. Decades later, these foundations are still recognizable (e.g., the hierarchical Bayesian approach of Latent Dirichlet Allocation in [2]). 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 will explore modern techniques for language processing, including `word2vec`, `doc2vec` and related technologies, and older tools like LDA. It will bring 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 and syntactic qualities of various Reddit.com communities will be compared. Reddit is one of the most-trafficked domains on the internet, with thousands of interconnected message boards and millions of submissions per day[3]. 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.

Reddit is organized like a forest graph: anyone can found a *subreddit*, the root of a tree. To this subreddit, anyone can post a *submission*, the first level of branches. These submissions can be blocks of text or links to other websites (often images or videos). On each submission, 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.

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 googling. Instead, this project will cluster subreddits

[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.

based on their comment sections in an effort to find communities similar to a target subreddit. This is an unsupervised classification learning 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 at a task (measured by a measure) improves with experience, as cited in [6]. Here, I will cluster subreddits (the task) learning only from the text of the comment sections of those subreddits (the experience), and will assess this clustering according to a similarity score (the performance measure, see Section 6). In the process of feature engineering, I will also explore classifying subreddits (using a usual classification scorer like accuracy) using the text of the comment sections as well as their subreddit labels.

3 Datasets and Inputs

Reddit provides an API for accessing its data, and this is the best way to get 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, some users have made available massive monthly data dumps of Reddit comments on the Google cloud. This source will provide the corpus for training and testing the model, using the Google BigQuery platform to filter on time, subreddit, and score.[3] The data will then be exported for local processing.

The exact SQL query used to extract the data is shown in Listing 1. It selects comments on three slices: time (from May 2015), subreddit (the top 100 most active subreddits) and score (the top 20,000 comments in each subreddit), for a total of two million examples. The historical data is chosen over more recent data due to memory constraints in BigQuery while sorting comments by score.

After this ETL process, the data consists of a large corpus of data points with two features: a raw block of text, and a subreddit label. All other information has been stripped for simplicity. Usable features to predict subreddit (and eventually, similarity to other subreddits) will be extracted from the comments using the principles of Natural Language Processing (NLP). K-fold cross-validation will be used during supervised feature evaluation and hyperparameter optimization (the scikit-learn implementation maintains class balance in the train-test splitting).

4 Solution Statement

The problem will be approached in two steps: (1) feature extraction from the subreddits and (2) clustering based on those features. In particular, only the comments of subreddit submissions will be considered (which assumes the culture of a subreddit exists in conversations around submissions, not in the submissions themselves). The comments will be analyzed using some natural language processing tools, and the results of that analysis will be used to find clusters of subreddits. For examples of algorithms to be used, see Section 7: Project Design, below.

5 Benchmark Model

5.1 Feature Extraction Benchmark

The simplest model for the supervised problem 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[8]). 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.

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

A more modern alternative is `FastText` from Facebook Research[4]: This extremely fast method has proven to be competitive with much more complex models at the forefront of contemporary research.

5.2 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. Due to the high-dimensional, sparse output of the frequency matrix used in Listing 3, we perform a simple feature selection using the singular value decomposition.

6 Evaluation Metrics

6.1 Feature Extraction Evaluation

Due to the inherently subjective nature of unsupervised learning problems (if we had an objective, it would no longer be unsupervised!), this project will attempt to compare feature extraction methods using a supervised learning problem, then apply clustering algorithms on the extracted features. Supervised evaluation metrics are much easier to come by: accuracy is defined to be the number of correct predictions divided by the total number of predictions. Furthermore, 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.

6.2 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 gold standard, 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. We want cohesion and separation; 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 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 cohesive 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 will instead use 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 cluster a_i, and the average distance

[4] `https://github.com/facebookresearch/fastText`

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).

6.3 Other Evaluations

This project will also consider 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 speed for both feature extraction and the actual clustering.

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

7 Project Design

The project will proceed in two phases. First, feature extraction tools will be compared using a supervised problem: given a corpus of comments from popular subreddits, compute the likelihood that a new comment belongs to any given subreddit (or simply assign each comment to a subreddit, as no certificate for regression exists in this data). Visualization and qualitative analysis of extracted features will also be performed. Once an appropriate set of features is identified, the unsupervised problem will transform comments from a particular subreddit and then find similar subreddits. More complex recommender systems will not be considered.

The project will explore and compare many modern tools for feature extraction:

- Bag-of-Words count matix, including n-grams
- tf-idf frequencies (nonlinear transformation of count matrix)
- Latent semantic analysis (SVD of count matrix)
- Latent Dirichlet allocation (Bayesian extension of a probabilistic LSA)
- `word2vec` (deep learning tool for dense word embeddings).

Some of these are tools for word embedding: mapping each word of a document to a point in some vector space. There are several options to aggregate the embeddings of each word in a document:

- Simple averaging techniques
- Training a convnet from scratch on `word2vec` inputs (note that the final dense classifier layers may be superfluous; here we are only considering feature extraction)
- `doc2vec` (deep learning tool for dense sentence embeddings)
- Transfer learning from pre-trained deep networks.

These feature extractors will be tuned and then benchmarked for time and accuracy using a few popular classifiers, including

- Multinomial Naive Bayes

- Approximate nearest neighbors (e.g., `Annoy`[1])
- `xgboost` or other vanilla classification methods
- Dense and shallow neural networks (especially for the deep learning feature extraction techniques like `doc2vec`).

Finally, clustering on the features will be performed using a broad spectrum of clustering techniques, including

- K-Means and variants
- Distribution-based mixture models
- Density-based methods like DBSCAN
- Adaptations to big data, like subspace clustering.

Parallelization and other big data techniques may be explored for some of these tools, as time allows. Most of these tools are easily implemented in `scikit-learn`[9], `gensim`[10], or `keras`[4].

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.

```python
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] 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.

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

[4] François Chollet et al. *Keras*. https://github.com/fchollet/keras. 2015.

[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] 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.

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

[10] 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.