Machine Learning Engineer Nanodegree
Capstone Proposal

Lee Burke

November 25th, 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*[6]. Decades years later, these foundations are still recognizable (e.g., the hierarchical Bayesian approach of Latent Dirichlet Allocation in [2]). The data revolution (see, e.g., [5]) 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 current Kaggle.com playground competition to identify the authors of horror story snippets [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 can comment on other comments. The conversations happening on Reddit occur almost exclusively in the comments: 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 only the comments saying, "cat.", are allowed). 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 based on their comment sections in an effort to find communities similar to a target subreddit. This is an

[1] `https://techcrunch.com/2017/05/14/building-a-smarter-hacker-news/`
[2] `https://www.kaggle.com/c/spooky-author-identification`

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.

I will extract relevant (and quantifiable) information from the subreddit comment sections (simple blocks of text, with the tree structure flattened). Then I will construct a supervised classification problem matching comments to their subreddit name in order to evaluate the power of the extracted information to find similarity. These features will then be used to suggest similar subreddits to subreddits not in the evaluation corpus.

3 Datasets and Inputs

Reddit provides an API for accessing its data, and this is the best way to get small amounts of up-to-date data. However, Reddit enforces a 30 request per minute limit, so trolling through millions of posts may be difficult. An example using the Python Reddit API Wrapper (PRAW) is shown in Listing 1.

Listing 1: PRAW example

```
import praw
## Store credentials in secrets.py
from secrets import *
reddit = praw.Reddit(client_id=CLIENT_ID,
                     client_secret=CLIENT_SECRET,
                     user_agent=USER_AGENT)
comments = [ comment for comment in
        [ submission.comments.list() for submission in
        reddit.subreddit('AskReddit').hot(limit=5) ]]
```

To avoid the significant time required to download a large number of these comments (due to rate limiting in the API), the website pushshift.io offers massive monthly data dumps of Reddit comments. This source will provide the corpus for training and testing the model, using the Google BigQuery platform to filter on time and subreddit; see `https://pushshift.io/using-bigquery-with-reddit-data/`. The data will then be exported for local processing.

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). Stratified K-fold cross-validation will be used during the supervised feature evaluation step.

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

The simplest model 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[6]). A simple implementation of this method with multinomial Naive Bayes is shown in Listing 1, where `X` contains comments and `y` is a label encoding of the subreddit names.

A more modern alternative is `FastText` from Facebook Research[3]: This extremely fast method has proven to be competitive with much more complex models at the forefront of contemporary research.

Listing 2: Benchmark model

```
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)
```

6 Evaluation Metrics

The obvious metrics are cost and accuracy: we want to classify the comments correctly as cheaply as possible. Cost will be measured as speed for both transformation/training and predicting, while accuracy will be 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.

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 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 this is a classification and not a regression problem). 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

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

- 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 will be stripped; 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`).

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`[7], `gensim`[8], or `keras`[4].

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] 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/`.

[6] 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`.

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

[8] 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.