Comparison of K-Means Clustering on Poetry and Spam Emails

Jack Rockwell, Judy Jackson, and Toby Harvey

1. INTRODUCTION

The application of machine learning techniques towards problems of natural language processing and generation has had exciting success when compared to previously exercised, traditional algorithmic methods. With the rise of the internet and the consequent explosion of availability and production of written content, there is more textual data out there than ever before. The development of computational tools to perform various functions on this massive and quickly growing body of literature will be vital to ushering in a new era of data-driven, macroscopic cultural studies.

In our explorations we've run into a diverse assortment of problems in the field of machine-learning-enabled natural language processing. On the level of implementation, we've approached two fundamental ones: automatic categorization of text by k-means clustering and automatic generation of learned textual models through a recurrent neural network. Implementation of solutions to both of these problems presuppose solving another, no less complicated problem, that of vector representation textual data. We've run them on two very different textual datasets: one website's collection of English-language poetry and a massive trove of spam emails.

The intersections of the two experiments on the two data sets can serve to measure our performance in different ways. We predict that the success of clustering will be easier to measure on the spam, because of the many near duplicates in the data set. The only human groupings against which to compare the clustering of the poems are the way they were organized on the website from which we scraped them, which is a questionable method of grouping the poems in itself, and therefore provides little metric against which to weigh the success of our clustering on the poems (more on this in the following sections).

We predict that the success of the RNN text generation will be easier to measure cursorily on the poems we will be able to read individual produced poems and determine if they have any meaning. If they were to make sense, we may also be able to determine by a cursory analysis of the distribution of language across the generated poems if our clustering of them worked earlier. On spam, we have no idea whats going to happen. But we think the idea of generating spam is cool.

2. PROBLEM EXPOSITION

2.1 Problems and Proposed Solutions

The reach of our explorations in this project is somewhat wide, and therefore can be reduced to no singular problem. The problems can be organized as follows:

(1) Automatically categorizing large amounts of text in a meaningful way
   (a) Proposed solution: implementing k-means clustering
   (b) Sub-problem: minimize intra-cluster variance

(2) Automatically generating meaningful text based on learned examples
   (a) Proposed solution: implement a Tensorflow RNN generative neural network

Presupposing both of these problems is that of converting text documents into an internal vector-based representation.

Minimizing intra-cluster variance can be measured easily enough. Whether or not our clusters or generated texts were meaningful is less empirically measurable, although we hope to have pretty good reasons to discriminate in the effectiveness of clustering between spam and poetry.

2.2 Data

We constructed two datasets to test our implementations of the solutions to these problems on. One was a collection of English-language poems we scraped off of blackcatpoetry.com . We downloaded every single poem on the website, 2,347 in total, where they are organized by their genre, assumably assigned by a human curator.

The second collection of spam emails comes from non-password-protected disposable mail sites dispostable.com, tempmail.org, and mailinator.com. Judys professor Cynthia Taylor scraped the collection off of these sites for her in 2015 as part of a project on security issues in disposable email. The original collection was massive-over 5 gigabytes of text in several different languages-so we implemented a brief filtering algorithm to obtain 2,000 randomly selected English-language emails. We used the Python library langdetect, a port of Googles language-detection software, to determine the language of the emails while they were filtered.

3. CLUSTERING EXPERIMENTAL SETUP

The objective of k-means clustering is to minimize intra-cluster variance. In order to do so, we considered four parameters: starting position of centroids, number of clusters, distance function used, and vector representation of documents. By changing these parameters, we hoped to find definitively better clusters for both the poetry and spam data sets based on their intra-cluster variance. Since the poetry data set came with human-given labels, we also assessed the similarly between the clusters given in the data set and the clusters generated by our implementation of k-means (although how meaningful the human-given labels actually are is questionable, a problem well return to).

3.1 Random Seeds for Centroids, and Number of Clusters

Since K-means is a expectation maximization algorithm, where centroids are picked randomly to initiate the algorithm, the intra-cluster variance can get stuck in a local minimum based on these initial guesses. In order to avoid this problem, we minimized the intra-cluster variance by implementing a grid search over five randomly generated seeds. For each randomly generated seed on the poetry data set, we ran our implementation of k-means with k in the range of 2 to 100, and 2 to 200 on the spam data set. We purposely

chose to run the poetry data set on a lower range of clusters for two reasons. First, because there are close to no repetitions in the poetry data set, so it seems reasonable to lower the number of clusters as we are not attempt to capture clusters of entirely repetitive poems to minimize intra-cluster variance. Second, while the labels in the poem data set do not represent a ground truth, as clustering is a unsupervised task, the fact that the data set has only 44 labels based on genre may be informative as to around how many genres we can cluster the poetry set into. We chose a larger range for the spam data set, because we noticed that it had a large number of nearly identical documents, some with entirely the same content except for a link, a word or two. It seemed plausible that if we allowed for enough clusters, we would reach a new global intra-cluster variance minimum, in which each cluster would solely contain emails that had near-identical content.

3.2 Distance Functions

In order to assign documents to clusters we needed a way of calculating the distance to the centroid of each cluster. Euclidean distance is the standard distance function used in k-means, although we also tried using the cosine distance function for varieties sake. Even though we had no hypothesis as to which distance function would yield a lower intra-cluster variance, it is of note that cosine distance is not dependent on vector length and only on orientation, while euclidean distance is dependent on both. What effect this has on the character of each cluster in the vector space created by our choice of vector representation we do not know.

3.3 Vector Representation

Vector representation is an important parameter, because if we want the documents in each data set to be clustered in distinct way, then we need each document to be represented in a way that brings out these distinctions when compared using a distance function. We‘re of the opinion that clustering is only useful if we can tell how the clusters are delineated. In other words, while there may be no ground truth value as to which document belongs to which cluster, a set of clusters is only useful if we can discern why a certain vector belongs to that cluster. This problem motivates intentional vector representation. We attempted to tackle this problem ourselves by creating a vector model using term frequency inverse document frequency, as well as making a model using the doc2vec python module.

3.3.1 *Tf-idf Model.* Term frequency inverse document frequency is a measure of the difference between how frequent a term in a document is and how frequent a term is in the entire corpus of documents, where:

$$\mathrm{tfidf}(t, d, D) = \mathrm{freq}_{t,d} \cdot \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

t is the term, d is a document, and D is all of the documents in the data set. We constructed vectors using tf-idf by setting each vector‘s length to the number of unique words in the entirety of data set, and calculating the tf-idf for each word in the document. The value at the index in which each term in the document occurred is then set to the tf-idf of that term in that vector-document. One serious downside to this representation of the documents is that the vector size grows with the addition of different words to the data set. The vectors will be huge with any good sized data set, increasing run time. Another is that these vectors are extremely sparse, because each document only contains a small amount of the words from the entire corpus. For an efficiency standpoint this is a very bad vector representation, and for that reason we only ended up using it on one k-means run-through on the poetry data set.

3.3.2 *Doc2vec Model.* As a better alternative we used the doc2vec python module, which trains documents on a neural network using word vectors, and accounts for word ordering unlike the bag of words model that our vectors used.

3.4 Cluster Similarity

As explained in sections 4.3 and 4.1, part of our goal was to create clusters that had some obvious characteristics to anyone reading through a few of the documents in each cluster. We wanted our clusters to be recognizable. Genre of poem is a recognizable way clustering, and though anyone can dispute what actually makes a genre (lack of a ground truth), there are still soft truth values as to what poem is in which genre. For instance, we can argue about whether or not ”A Supermarket in California” by Allen Ginsberg is in the ”supermarket poetry” genre or ”the beat poets” genre, but we would both agree it is not part of the ”beach poems” genre. With the caveat that genre is a fluid distinction, to measure how well our clustering did in determining genre we also calculated the Adjusted Rand Index. Adjusted Rand Index (ARI) is a similarity that takes on values on [-1,1] where -1 and 1 represent identical clustering, and 0 represents opposite clustering. We calculated ARI on our poetry clustering, but did not on our spam as we didn‘t consider any other recognizable clustering for that dataset.

4. GENERATION EXPERIMENTAL SETUP

For the generative aspect of our project, we implemented a character-based Recursive Neural Network using code from https://quirkyai.wordpress.com/2017/06/07/text-generation-in-lines-with-lstm-rnn-in-python-tensorflow/. The program [textgenA.py] reads in a single text file, converts all the characters to their integer values, and produces a training set of sequences with length specified by the user (we used a sequence length of 100). We kept the default values for the Long Short Term Memory (LSTM) metric and the keep rate for purposes of efficiency. For each input text, the program created a model from 20 epochs of learning, each with a batch size of 250. The network applied a softmax function as its activation function and used a categorical cross-entropy function for the loss/error function when calculating the regressions. Because the code relied on several pre-existing libraries, our main addition to the code was tweaking the values of various parameters. We found that the models generated faster with a larger batch size. We kept the sequence length at 100 characters with the rationale that this length was long enough to capture word dependencies and small enough to have enough variation with larger training samples. To generate text, we reloaded the same model, picked a random sequence from the training set, and generated a sample based on that sequence as a seed.

5. RESULTS

Our overall results add a little evidence to our hypothesis that spam is significantly easier to cluster than poetry is, but also are mainly inconclusive. We first explain our results for spam, then poetry, then compare and discuss them.

5.1 Spam Results

We first grid searched as described in 4.1 using cosine distance and doc2vec on the spam data set. We determined how good a cluster-

ing was by first asking if it was an elbow point in the intra-cluster variance vs. number of clusters graph (if not it was excluded from the results), and then what its actual variance was. Figure 1 shows the 5 graphs created for all different values of k on each random seed. We determined the elbow point of each graph by visual inspection alone. They are marked in red. Obviously the elbow points with higher number of clusters have a lower intra-cluster variance, but we still include both elbow points, as any significant change in the rate of decreasing intra-cluster variance could equate to an important clustering. The lowest intra-cluster variance achieved was on seed 30 with 172 clusters (the second elbow point), a euclidean distance function, and a doc2vec vector representation. Many of the clusters in this clustering were composed of over 50 percent of nearly-identical documents, and some consisted of only nearly-identical documents. This result implies that if we did not let the number of cluster range large enough to create completely homogeneous clusters. While a completely homogeneous clustering would probably be the global minimum, these clusters be so trivial they would completely miss the point of clustering. Who wants a cluster of only the same thing? After further examining a few clusters, it becomes fairly obvious that the clusters that are not homogeneous are not categorized by any recognizable genre. For instance cluster number 7 contains spam about password resets, working out, and creating accounts on gaming websites.

5.2 Poems Results

5.2.1 *Clusterings without number of clusters equal to number of labels.* We used the same grid search technique as above to find optimal clusters which yielded five graphs that appear nearly identical. We averaged all intra-cluster variance values of each number of clusters, and re-graphed the averages in Figure 2. Figure 2 does not appear to have any particularly obvious elbow points, so instead we fit a line to the data, and found the point of maximum curvature (the closest point we could think of to an elbow point). Curve fitting in mathematica gives us the equation:

$$V(k) = 7192 + \frac{190814}{k} - 91k$$

Then using the formula for curvature:

$$\kappa = \frac{V''(k)}{(1 + V'(k))^2}$$

and maximizing on [0,100] gives a maximum curvature at k = 42. When running our implementation of k-means on the poetry with 42 clusters on one of the seeds used in the earlier averaging (45) and doc2vec vector representation, euclidean distance outperforms cosine distance again, with a intra-cluster variance of 6,607. This is significantly higher than our best results on the spam. Similar to the spam data set there is no discernible theme within many of the clusters. For instance, some of the poems in cluster 8 are about medieval war, some are about modern day New York, while still others in are in-depth descriptions of trees and clouds.

5.2.2 *clustering with same number of cluster as labels.* Running our implementation of k-means with the same number of labels as in the poetry data set allows us to calculate the ARI between our clustering and the website's. Over 5 random seeds with both euclidean and cosine distances, the ARI ranged from .02 to .036, which pretty conclusively means our clustering of poetry was nothing like the websites categorization of them by genre.

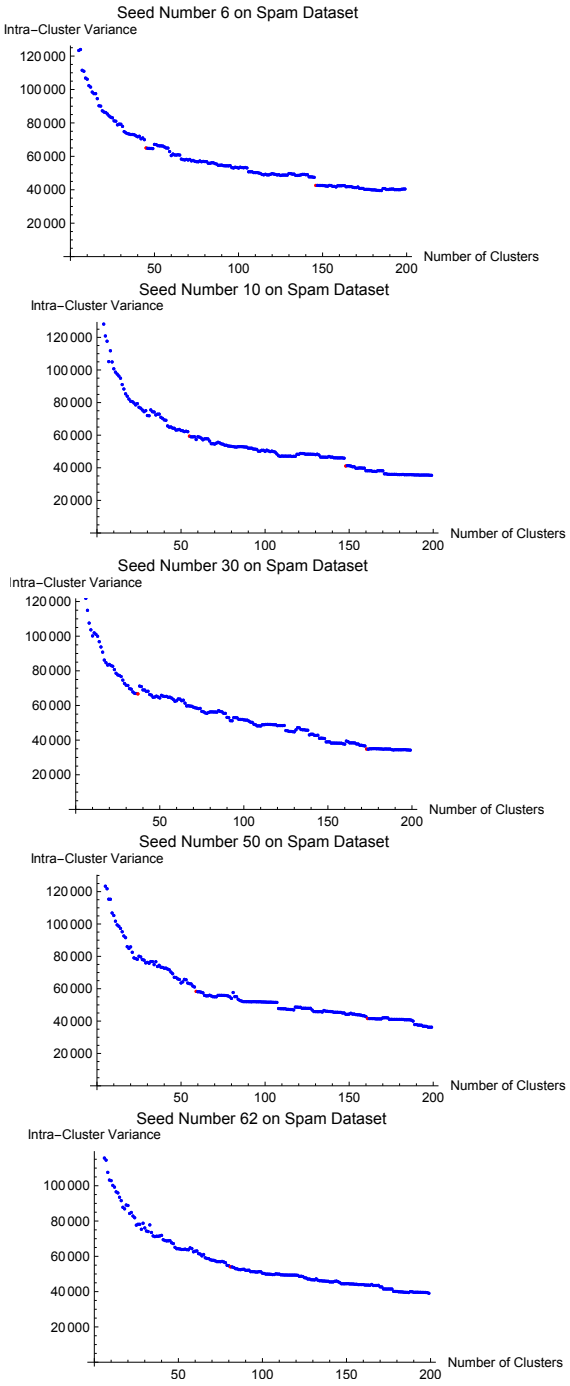

Fig. 1. intra-cluster variance as a function of number of clusters

5.2.3 *clustering using our tf-idf vectors.* We ran one round of k-means using our tf-idf vectors, but the result a complete failure. The intra-cluster variance for this single run was 8,856,843, and the vast majority of poems went into a few clusters, while other clusters only contained a single poem. Such a high variance may be attributed to the fact that there are many poems in the data set that share only a few words, so that the vast majority of distances computed between components of vectors when finding the nearest

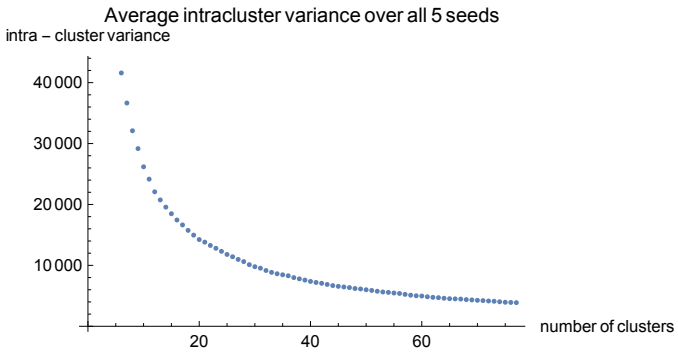

Fig. 2.

centroid are between 0‘s and tf-idf values, instead of between pairs of tf-idf values.

5.3 Discussion

5.3.1 *Bad Data Sets.* Our poetry clusters ultimately had no real cohesive themes within them and significantly higher intra-cluster variance than the spam clusters. The spam data set had significantly lower intra-cluster variance, but it had many near duplicates, and observing the spam clusters we saw that either they were completely homogeneous, or homogeneous with two or three types of spam in one cluster. This is a disappointing result, because it points to one explanation. Our version of k-means was only really identifying whether or not documents where nearly identical. In this case the only real reason why spam seemed easier to cluster than poetry was that the spam data set had duplicates. If we instead had used a spam data set with no repetition we would have found the similar results as on the poetry data set.

5.3.2 *Drop in intra-cluster variance.* The drop in intra-cluster variance when switching from our tf-idf vector format to the vec2words format does give some hope that we could cluster the poetry data set in some sort of coherent way, simply because k-means can found a way to tighter clusters.

5.3.3 *Euclidean vs. cosine distance.* The runs using euclidean distance pretty drastically out preformed the runs using cosine distance. On the spam data set the clustering using euclidean had variance around 2000 ± 1000, and there counterparts using cosine distance had variance 50000 ± 2000. We don’t have a good explanation for this, and it seems possible there is a bug in our code. A in depth exploration of the properties of distance functions on unorthodox vectors spaces like the ones we were working in may shine some light on this difference too.

6. GENERATION RESULTS

The generative model did not work particularly well: even with the spam dataset, it only produced a repeated character (i.e. eeeeeeeee...eeee). Generating the model using more epoch iterations may have produced better results, but each model with 20 epochs took several hours to generate, limiting our capabilities. We found that the loss values produced for each epoch for the poetry were less than those for the spam (spam started at around 3 and dropped to around 2.8, whereas poetry started at 2.2 and dropped to about 1.6). Based on this metric, it seemed that the network had an easier time learning the poetry than it did the spam. This could be because the spam emails tended to have various random characters and number/letter sequences (more variance in the training characters) as compared to the poetry, which contained fewer characters overall. Because the generative model failed to produce anything that resembled text, we were unable to check the accuracies of our clusters using the generation as a verification.

7. CONCLUSIONS

Our attempts to address the problems of meaningful categorization and generation of texts failed. What we did learn, however, is why they failed and what directions to look towards in future experiments.

The nature of our datasets prefigured our attempts to categorize them by k-means clustering towards failure. For the poetry data set, we‘ve determined that the relationship between all forms of numeric vectorization of poems that we tried and human-assigned genre labels is practically non-existent. We could discern, by reading of the grouped poems, no other relationship between poems within the same clusters. For the spam data set, the near-duplicity of many of the emails threw a wrench in our clustering, for, as we said, we don‘t need a complex k-means clustering algorithm to tell us that a bunch of emails are almost exactly the same. Even looking past the clustering of duplicates, there were no discernible relationships between the contents of the types of emails that were within the same clusters. It seems that for now, k-means clustering by the numeric vector representations of texts we used does not approximate human discrimination of texts by meaning.

As far as generation is concerned, our failure to successfully generate more than one repeated character makes any analysis of generated poetry or spam emails impossible.

Was it worth it? Absolutely. We had a lot of fun learning about this stuff, even if we didn‘t get everything right, and we would do it all over again. It was nice to imagine that unsupervised learning would result in some sort of recognizable categorization of text. It was also nice to speculate as to what generated poems and spam might look like, and how the appearances of these texts might differ and/or converge. Before recommending any further directions for research, we recommend someone duplicate our experiment without near-identicals in the spam and functioning generation.

REFERENCES

Quoc Le, Tomas Mikolov. *Distributed Representations of Sentences and Documents.* http://arxiv.org/pdf/1405.4053v2.pdf. Google Inc, 2014.

Lawrence Hubert, Phipps Arabic. *Comparing Partitions.* https://link.springer.com/content/pdf/10.1007%2FBF01908075.pdf Journal of Classification 2:193-218 (1985).

quirkyai. https://quirkyai.wordpress.com/2017/06/07/text-generation-in-lines-with-lstm-rnn-in-python-tensorflow/