CS 401R Final Project

Andrew Merrill and Jacob Stern

April 2018

1 The Data

1.1 The Dataset

Our dataset consists solely of the text of the Book of Mormon. The plain text of the Book of Mormon was found at http://www.gutenberg.org/cache/epub/17/pg17.txt.

1.2 Importance of the Dataset

This text analysis has importance to us because we are very interested in being able to find common themes in the scriptures and where they are found. The idea is to be able to enter in a section of scripture and have the algorithm return a list of scriptures that are very similar. Using this, we can find where themes are repeated throughout the Book of Mormon. We imagine this could benefit others in their study of the scriptures.

2 The Problem

2.1 Description

The idea we had was to analyze the text of the Book of Mormon and find common themes that arise. We were intrigued by the idea that there are themes and topics that one prophet teaches that are also taught, in a slightly different way, by other prophets. We learned about the idea of word embeddings, which we found very interesting. We decided that we wanted to learn how to use word embeddings in order to vectorize the text of the Book of Mormon and search for clusters of verses or sentences by topic. This is an unsupervised classification problem, with the goal of classifying a set of verses, and finding verses that are similarly classified.

2.2 Our Background

The background knowledge we brought to this project was simple, 2-dimensional Expectation Maximization. This problem added a level of complexity – our goal was to work on 300-dimensional vectors instead of 2-dimensional vectors.

2.3 Other Approaches

We could not find any other examples of people using expectation maximization with word embeddings to cluster topics in a document. There were examples of people analyzing the text of the Bible, but it was more focused on word frequencies, and finding lists of words that are similar to each other, etc.

An alternate approach that other people have taken to find word similarity is to compare the similarity of the angle between two vectors. In our approach we wanted to compare the entire verse vectors, but an alternate approach could be to take the angle between verse vectors:

$$\theta = \cos^{-1} \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||||\mathbf{b}||} \tag{1}$$

3 Exploratory Data Analysis

3.1 Organizing the text

In order to convert the words in the Book of Mormon into vectors, we first had to clean up the text by removing punctuation and any other non-alphabetic characters and organize the text into a format we could work with. We put it into a Python dictionary, with the keys being the names of the books, and lists holding chapters and verses. This would allow us to quickly access specific verses in order to vectorize them. The user could input a set of verses, and the algorithm could grab them and compare them to other verses to find similar topics discussed in other places.

There are further explorations of the data, along with visualizations and interesting trends in section 4.2.1.

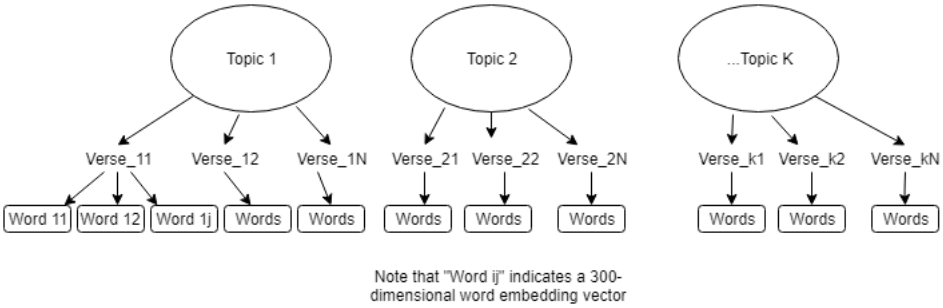
4 Description of Technical Approach

4.1 Background on the Approach

We did not find much research that described how word embeddings can be used, other than for simple word similarity comparisons. We hoped to find a way to use expectation maximization to cluster similar groupings of words (in this case, verses) using word embeddings.

4.2 Description of the Model

We model verses by assuming that each verse was generated by sampling words from a topic. Each verse is assumed to have one topic. The topics are modeled as a distribution over word embeddings. The distribution of topics over word embeddings is represented as a 300-dimensional Gaussian Mixture Model, where each topic has a 300x1 mean vector and a 300x300 covariance matrix.



4.2.1 Working with Word Embeddings

Choosing a Word Embedding Model

In order to vectorize a word, you need to train a model on a corpus of text so that it can learn the relative meaning of words to each other. We first heard about Word2Vec, and started with that. We found a pre-trained model from Google that was trained on a corpus of three million words from Google News. We loaded this model in and began testing it with the text from the Book of Mormon. We quickly realized we had a problem with the language of the Book of Mormon. Words ending in "eth" or "est" like "endureth" or "liest" don't exist in the Google model. There were 230 unique words that fell into this category. On top of those, there were about 100 unique words that were out-dated words not showing up in the model. Lastly, there were 163 unique names from the bible that did not show up in the model. We included a visualization of these unknown words to show some specific examples. From here, we decided to try a new model that could hopefully include these older words. We decided to train our own model, and as our corpus we used a text file of 11.9 Gigabytes of Wikipedia articles. However, when we ran similar tests on it, it ended up being worse for vectorizing the old language. We also tried training it with both the Wikipedia articles as well as the full text of the Bible, along with other novels of similar language available in a Python library: nltk.corpus.gutenberg. This didn't prove to help any. We are assuming this is because the massive volume of Wikipedia articles prevented any of these uncommon words from making it into the model. We decided to use the original Google model.

Modifying the Text to Fit the Model

From here, we decided to modify the text itself in order to modernize words and find a vector for as many words as we could. We started by removing 'eth' and 'est' from all words. This ended up removing a lot of those words, but it didn't quite work for all words to just remove it, so we created a mapping so that all words would be edited correctly. An example would be 'twentieth'. You have to leave the 'eth' or it is not a word.

We also removed stopwords from the text. These are words like 'am', 'by', 'a', 'the', etc. that do not really have a lot of meaning in the text, so if you remove them you can focus on vectorizing words that contain the real meaning of the verse.

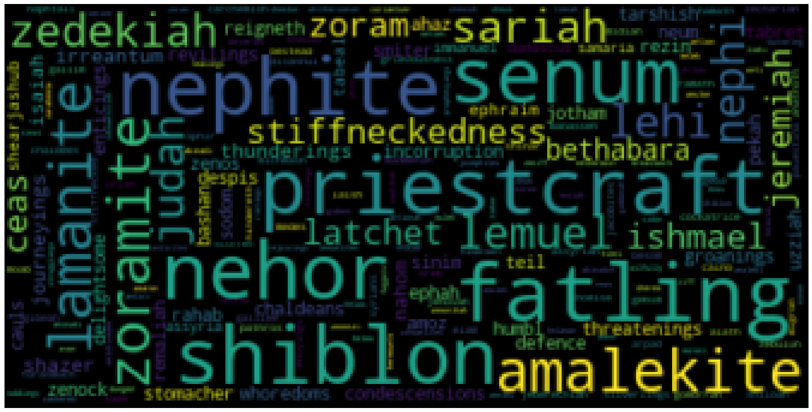
This ended up leaving 371 unique words, all of which were old words and names. If we had more time, we could try to map all of these words to their more current counterparts, and we could try to map the names to more current names or to pronouns.

With these remaining words, we did some research and found that one alternative was to take the average of the surrounding words, and use this as the vector for the word that doesn't exist in the model. We wrote a function to take the average of the words around each of these, and ran through the whole Book of Mormon again. This worked for the majority, but we had another problem come up. We had it set up to take the average of the two words before and the two words after, and there ended up being some cases where there weren't any words around it that could be used to average. We printed out the problem words and their corresponding verses, and the results were interesting. There are verses where there are lists of words all in a row that don't exist in the model. Thus, when trying to average the surrounding words, none of them have vectors, and it cannot create a new vector for the word. We included a picture of a couple examples of these verses. Considering the time we had, at this point, we decided to leave these verses alone. We would perform testing on the verses that we had fully vectorized, and when we have time we will work on vectorizing these outlier verses.

Final Decision on Word Embedding Model
This model ended up not being completely ideal. Moving forward, we would like to try a word representation library called fastText. It has the ability to look at parts of words in order to find the meaning of a word. We haven't tried it, but the idea would hopefully be that it would recognize the words ending in 'eth' and 'est' because the root of the word is the same. This would be able to identify more words, and we wouldn't have to try to create artificial vectors for so many words.

4.2.2 Visualizations

Frequency of Words That Did Not Fit into the Model



Examples of Verses with Words That Could Not Be Vectorized

```
2 Nephi 19 manasseh ephraim ephraim manasseh they together shall be against judah for all this his anger is not turne
d away but his hand is stretched out still
2 Nephi 19 manasseh ephraim ephraim manasseh they together shall be against judah for all this his anger is not turne
d away but his hand is stretched out still
2 Nephi 19 manasseh ephraim ephraim manasseh they together shall be against judah for all this his anger is not turne
d away but his hand is stretched out still
Jacob 1 now the people which were not lamanites were nephites nevertheless they were called nephites jacobites joseph
ites zoramites lamanites lemuelites ishmaelites
Jacob 1 now the people which were not lamanites were nephites nevertheless they were called nephites jacobites joseph
ites zoramites lamanites lemuelites ishmaelites
Jacob 1 now the people which were not lamanites were nephites nevertheless they were called nephites jacobites joseph
ites zoramites lamanites lemuelites ishmaelites
Jacob 1 now the people which were not lamanites were nephites nevertheless they were called nephites jacobites joseph
ites zoramites lamanites lemuelites ishmaelites
Jacob 1 now the people which were not lamanites were nephites nevertheless they were called nephites jacobites joseph
ites zoramites lamanites lemuelites ishmaelites
Mosiah 6 and mosiah began reign in his father s stead he began reign in the thirtieth year his age making in the whol
e about four hundred seventy six years from the time that lehi left jerusalem
Mosiah 7 and ammon took three his brethren their names were amaleki helem hem they went down into the land nephi
Mosiah 8 and ammon said that seer is revelator prophet also gift which is greater can no man have except he should po
ssess the power god which no man can yet man may have great power given him from god
Mosiah 19 and gideon pursued after him was about get upon the tower slay the king the king cast his eyes round about
towards the land shemlon behold the army the lamanites were within the borders the land
Mosiah 23 and amulon his brethren did join the lamanites they were traveling in the wilderness in search the land nep
hi when they discovered the land helam which was possessed by alma his brethren
Mosiah 24 and amulon commanded them that they should stop their cries he put guards over them watch them that whosoev
er should be found calling upon god should be put death
Alma 10 and aminadi was descendant nephi who was the son lehi who came out the land jerusalem who was descendant mana
sseh who was the son joseph who was sold into egypt by the hands his brethren
Alma 11 a senum silver an amnor silver an ezrom silver an onti silver
Alma 11 a senum silver was equal senine gold either for measure barley also for measure every kind grain
Alma 11 and shum gold was twice the value seon
Alma 11 and limnah gold was the value them all
Alma 11 a shiblon is half senum therefore shiblon for half measure barley
Alma 11 and shiblum is half shiblon
```

4.3 Description of Training Algorithm: Expectation Maximization

We used expectation maximization, an algorithm that attempts to refine a model by finding the parameters for the model that maximize the likelihood that the

model generated the data. In this case, those parameters are the mean vectors and the covariance matrices of the clusters.

In order to get our algorithm working, we utilized our generative model to create well-clustered data. We tested our algorithm on that data to see if it would be able to successfully cluster the data.

4.3.1 Adjusting the Expectation Maximization Algorithm for High-Dimensional Data

We worked to cluster 300-dimensional vectors. This resulted in complications when attempting to visualize the data. One method of visualizing high-dimensional data is Principal Component Analysis (PCA), which takes the aspects of the data that give the most information and visualizes those. We didn't have time to implement this tool, but instead plotted the first two components of each vector, which generally worked well for well-clustered data.

4.3.2 Complications: Covariance Matrix turns Non-invertible

The Problem: When evaluating the probability density function to compute responsibilities for each sample, scipy threw a "singular matrix" error. The problem was that our covariance matrix was non-invertible. Dr. Wingate suggested adding a trivial multiple of the identity matrix to the covariance matrix so that the matrix would become invertible. We tried that, and it didn't work. We were still getting a "matrix must be positive semi-definite" error. In doing some more digging online, we learned that such an issue often arises when the eigenvalues of the covariance matrix end up very slightly negative (on the order of magnitude of 10^-16). However, adding values to the diagonal should resolve that. So we tried printing the minimum eigenvalues of the matrix. About half the time, the smallest eigenvalue was about what you would expect – around -1e-14. But the other half of the time, the minimum eigenvalue was a whopping -377.80, or something similarly huge. We realized that the problem was that earlier in the program, we were adding Gaussian noise to the expected covariance matrix, which resulted in a covariance matrix with hugely negative eigenvalues that was no longer invertible. After removing the noise and adding a small multiple of the identity matrix, we were successfully able to evaluate the PDF for our data.

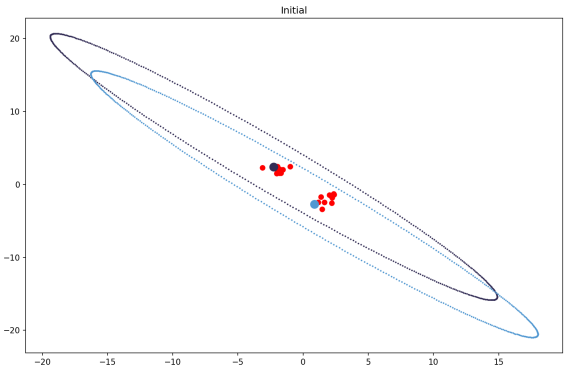
In researching these errors, we discovered that the problem of non-invertible covariance matrices is a common one, and learned about several solutions – from adding a trivial value to the diagonal, to resetting the covariance to its previous value, to factoring the covariance matrix to make it easier to work with.

4.3.3 Working with the EM algorithm

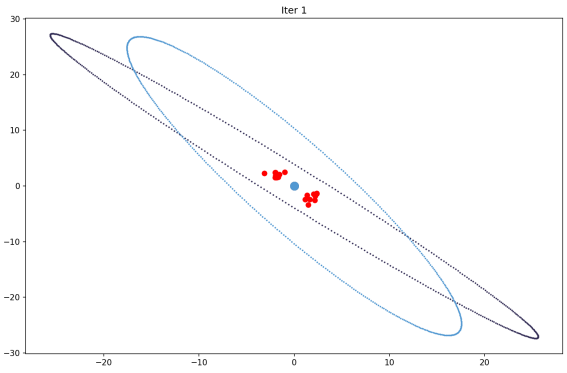
The next problem was that as soon as we ran an iteration, the means of our clusters collapsed so that they were essentially right on top of each other.

It is a common problem of high-dimensional data to overfit. What can happen is that one cluster comes to represent a single point, and the covariance of one cluster collapses so that this cluster represents perfectly that one point. This results in a likelihood of near-infinity, but poor clustering.

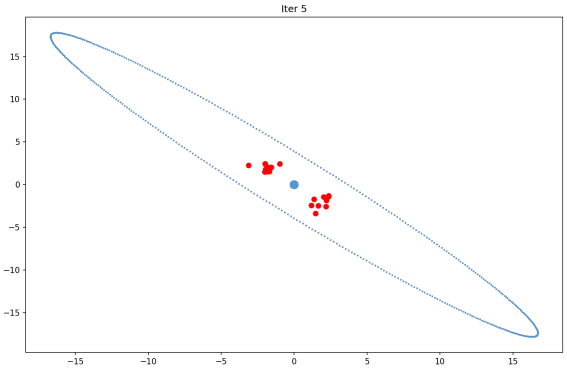
The strange thing is that this is not what happened with our data. Instead the clusters merged with each each other and ended up with exactly the same mean and covariance. It will take further research to figure out why that is.



Initial guesses for clusters



Means quickly collapse



Covariances quickly follow

4.4 Partitioning Data into Test/Training Split

We are using an unsupervised algorithm, but to see how well it was doing at clustering the data, we withheld several vectors from the set and tracked how well they fit into the clusters as time evolved.

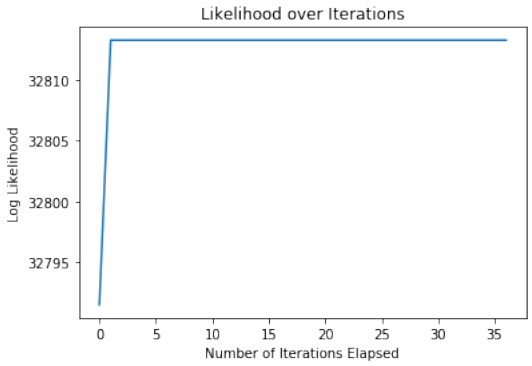
5 Analysis of Approach

5.1 An analysis of how your approach worked on the dataset

We didn't make it all the way to testing our algorithm on our dataset. We generated neatly clustered vectors with our generative model, but our algorithm was unsuccessful at discovering those clusters. If we were able to get EM working for 300-dimensional vectors, we could then work on clustering real word embeddings.

5.2 What was your final RMSE on your private test/training split?

We measured the optimization of our algorithm by the log likelihood of our data given the current parameters for our Gaussian Mixture Model. We summed the log likelihood of each observation (data point) at each iteration, and the results are displayed below.



5.3 Did you overfit? How do you know?

That is the real question! We believe that we did, and that is why our clusters end up right on top of each other. As previously mentioned, we are unsure why identical parameters for both clusters maximized our likelihood.

5.4 Was your first algorithm the one you ultimately used for your submission? Why did you (or didn't you) iterate your design?

EM seemed like the true machine learning approach, and so we stuck with it for almost all of the time. We did find some algorithms online that dealt with the complexities of higher-dimensional EM. If we were to do it again, we would take more time to study those algorithms and possibly use one of theirs in place of our own.

We decided at the last minute to try a new design where we compared the angle between each verse embedding vector with every other verse embedding, and chose the top ten nearest matches. This is a simpler method than EM, but turned out to achieve our same basic objective – to find groups of similar verses in the Book of Mormon.

5.5 Did you solve (or make any progress on) the problem you set out to solve?

We didn't solve our problem the way we intended to, due to the complications of high-dimensional EM. However, we made progress on working through some of the issues that arose. Ultimately, we spent a brief amount of time trying a new design that worked much better. We learned that sometimes the best thing to solve your problem is to let go of your original idea and try a new approach.

We tried running this algorithm on each verse, so that we would have the top ten verses for each verse, but with the time we had, it was a very naive approach

with no vectorization, so we only ended up having time to go through Alma chapter 26. When this was complete we searched through for the verses that found the top ten verses with the smallest angles. The top three candidates were as follows:

2 Nephi 33:1 0.44109
Angle: 0.45154 Verse: Mosiah 3:14
Angle: 0.44833 Verse: Jacob 4:13
Angle: 0.44215 Verse: Jacob 2:20
Angle: 0.44451 Verse: Jacob 4:12
Angle: 0.44097 Verse: Alma 32:4
Angle: 0.43916 Verse: Jacob 3:4
Angle: 0.43720 Verse: Jacob 6:3
Angle: 0.43043 Verse: 1 Nephi 1:20
Angle: 0.43600 Verse: 1 Nephi 13:28
Angle: 0.44059 Verse: Mosiah 2:40

Mosiah 21:13 0.42079
Angle: 0.42861 Verse: Alma 56:55
Angle: 0.42826 Verse: Mosiah 29:19
Angle: 0.42786 Verse: 1 Nephi 7:20
Angle: 0.42611 Verse: 1 Nephi 8:36
Angle: 0.42254 Verse: 3 Nephi 3:24
Angle: 0.42421 Verse: Alma 36:1
Angle: 0.41505 Verse: 1 Nephi 16:38
Angle: 0.41560 Verse: 2 Nephi 5:20
Angle: 0.41917 Verse: 1 Nephi 1:20
Angle: 0.40047 Verse: Mormon 3:2

Alma 5:10 0.42809
Angle: 0.44651 Verse: Alma 20:25
Angle: 0.44241 Verse: 2 Nephi 31:11
Angle: 0.44135 Verse: 1 Nephi 10:16
Angle: 0.43905 Verse: 1 Nephi 5:3
Angle: 0.43618 Verse: Ether 12:19
Angle: 0.42679 Verse: Enos 1:1
Angle: 0.42144 Verse: 1 Nephi 3:27
Angle: 0.42423 Verse: Alma 33:16
Angle: 0.37058 Verse: 1 Nephi 2:15
Angle: 0.43237 Verse: Ether 4:9

The number next to the top verse (the verse being compared with all other verses) is the average of the angles to the top ten verses. This acts as a score to see relatively how good the verses are that were returned. From the data we gathered, we found that the minimum was .37058, and the maximum was .45154.

This is a pretty narrow range using the naive angle comparison. However, from the verses I looked at from these best verses, it looks like it is still working to a degree. So either these are just coincidences or there are enough verses in the Book of Mormon, that there are ten verses similar enough to reach that level of similarity. We would like to think the latter is true. It is very interesting looking through the results.

So interestingly, the the verse from the first half of the Book of Mormon that is most similar to any other verse in the Book of Mormon, going strictly off of the angle of the verses vector, is Alma 5:10 compared to 1 Nephi 2:15.
Alma 5:10 And now I ask of you on what conditions are they saved? Yea, what grounds had they to hope for salvation? What is the cause of their being loosed from the bands of death, yea, and also the chains of hell?
15 And my father dwelt in a tent.

So, kind of a funny example. Here are a few that we found pretty interesting though.

Comparison verse:
Mosiah 21:13 And they did humble themselves even to the dust, subjecting themselves to the yoke of bondage, submitting themselves to be smitten, and to be driven to and fro, and burdened, according to the desires of their enemies

Similar verses found:
Mosiah 29:19 And were it not for the interposition of their all-wise Creator, and this because of their sincere repentance, they must unavoidably remain in bondage until now.
Alma 56:55 And now it came to pass that when they had surrendered themselves up unto us, behold, I numbered those young men who had fought with me, fearing lest there were many of them slain.
1 Nephi 7:20 And it came to pass that they were sorrowful, because of their wickedness, insomuch that they did bow down before me, and did plead with me that I would forgive them of the thing that they had done against me.

Comparison verse:
2 Nephi 33:1 And now I, Nephi, cannot write all the things which were taught among my people; neither am I mighty in writing, like unto speaking; for when a man speaketh by the power of the Holy Ghost the power of the Holy Ghost carrieth it unto the hearts of the children of men.
Jacob 4:13 Behold, my brethren, he that prophesieth, let him prophesy to the understanding of men; for the Spirit speaketh the truth and lieth not. Wherefore, it speaketh of things as they really care, and of things as they really will be; wherefore, these things are manifested unto us plainly, for the salvation of our souls. But behold, we are not witnesses alone in these things; for God also spake them unto prophets of old.
Alma 32:4 Now, as Alma was teaching and speaking unto the people upon the hill Onidah, there came a great multitude unto him, who were those of whom

we have been speaking, of whom were poor in heart, because of their poverty as to the things of the world.

This was a very fun project to work on. It was awesome to learn about word embeddings, multidemensional EM, and comparing word embeddings to find similarities. We want to do more work on this to improve it and continue learning more.