Jacob Dineen

Text Mining – IST 736

Homework 8

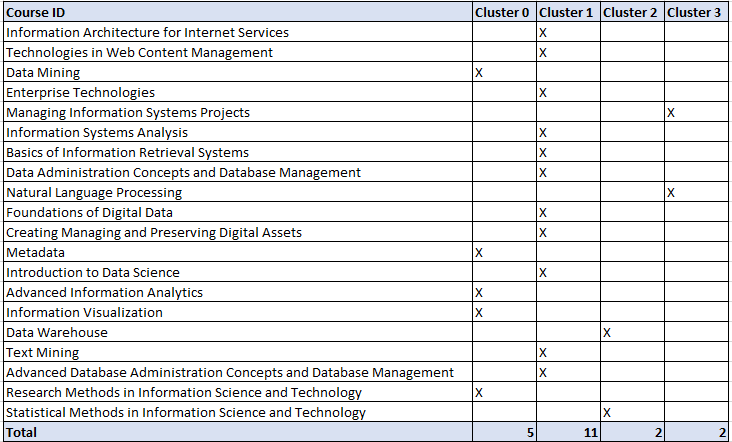
Task 1 – Document Clustering

**INTRODUCTION:**

Document clustering is a form of unsupervised learning, meaning that there is no implicit mapping from a set of inputs to a set of outputs. Clustering works by minimizing some distance function (L1/L2) between the centroid of a cluster and each point represented within that cluster in a high dimensional space. We can look at clustering as a form of minimizing the within-cluster variance while simultaneously trying to create as much distance as possible between our k clusters (between cluster variance). In this assignment, we are tasked with using Kmeans clustering to try and identify patterns between the required or elective courses in the CAS in Data Science, as offered by Syracuse University. We had previously discussed our preliminary results during the asynchronous material, but I’ll attempt to decipher what words, specifically, are driving some of the clustering decisions.

**METHOD:**

Weka will be the tool of choice here, mainly due to the accessible GUI and the ability to portray high dimensional representations graphically – Doing so in Matplotlib or Seaborn would take a bit of programming to be insightful. We previously read in the CSV file and saved it as an AARF, so there isn’t any file conversion necessary here, although we do need to convert our CourseDesc column into a string field from its current nominal state to be able to further convert it into a sparse representation. From here, we can implement a Weka filtered classifier which will be one part vectorization and one part k means clustering. While some took the approach of converting course ID into a feature, I thought that may be a form of algorithm manipulation – From my understanding of Kmeans Clustering, you might incidentally be enforcing an implied distance between clusters (We’re using Euclidean Distance here). I settled on K = 4 for two reasons. There appears to be a diminishing return in the loss function (with cluster SSE), and visualizing or the ability to interpret more clusters might lessen with the increased number of K. I

**RESULTS:**

Initial results show that we have broken the inputs (n = 20) into three bins, which plotted on a histogram may look like a right skewed distribution. 25% of observations fell into cluster 0, 55% to cluster, 10% into cluster 2 and 10% into our final cluster.

In a table to the left we can see the cluster that each course fell into, sorted in ascending order by course ID. This shows us that there is no natural separation in course level derived, and although we see some similarities between courses at a hierarchical level, it’s difficult to ascertain a pattern just by looking at this. Note that Lovin’s stemmer was used in our vectorization schema, and we didn’t utilize extract grams > 1.

Some insights into looking at our final attribute centroids by attribute/word:

Cluster 0: This cluster had a higher propensity to include the words/stems ‘introduc’ and ‘techniqu’, although most of these courses are an introduction to using some technique for analysis, so this is probably just a case of linguistic stylization of course descriptions.

Cluster 1: Cluster 0 and 1 were both more likely to include the stems ‘dat’, possibly referencing an inclination toward data-based courses. This also seems like an oddity because most of these courses are implicitly focused on data in some regard.

Cluster 2: Appeared to be separated, in part at least, on the stem of the word procedure.

Cluster 3: This cluster was more prone to having the words ‘technology’, ‘linguistics’ and ‘computation, although there were only two observations clustered together here.

**CONCLUSIONS:**

In conclusion, I recognize the importance of unsupervised learning in a practical sense, but I don’t think it was well suited for this task. I believe that the sample size was too low for us to draw any real insight out of what separates clusters – Most anomalies tied to differences in attribute centroids seemed more by chance than having been significant patterns uncovered. I would have expected to see some relationship between courses that were focused on Databases against machine learning against text analytics, but there was a stochastic nature to the findings.

Task 2 – Topic Clustering

**INTRODUCTION:**

Latent Dirichlet Allocation is a generative model in the unsupervised space that helps to explain some of the causal relationships between topics and features. The algorithm posits that [1] a collection of documents is made up of individual parts (topics/words) and that explain some of the separation between clusters. We are essentially trying to find the highest probability of selecting a specific word against a topic. As I’ve cut my time to work on this relatively short, I’ll be using some of the precompiled functions/commands lended to us in the Mallet documentation. Mallet is a statistical package that I am relatively unfamiliar with (in industry) but seems to have some practicality in many avenues of machine learning. For this task we will be looking at the main topics in the floor debate of the 110th Congress (House Only). The benchmark for topics, as expressed by political scientists, is 40-50 common topics against each Congress. We are to tune the number of topics and see if we can understand some of the parts that make up each topic.

**METHOD:**

To begin, we must convert our aggregation of documents into a .mallet file. We can do this by running:

*bin\mallet import-dir --input C:\Users\jdine\Documents\110 --output congress.mallet --keep-sequence --remove-stopwords --gram-sizes 1*

Here, we remove the stopwords and set gram-size to 2. Setting multiple n-grams breaks this command for some reason. I initially used unigrams, but the results were nonsensical

Onto topic modeling, we run the following command to run LDA on our documents:

*bin\mallet train-topics --input congress.mallet --num-topics 40 --optimize-interval 20 --output-state sample-topic-state.gz --output-topic-keys sample-keys.txt -–output-doc-topics sample-topics.txt*

This took ~17 minutes to run on a 6core processor.

**RESULTS:**

Analyzing topics via word or topic intrusion seems like a highly manual process with a collection this large. Nevertheless, below are some of my findings:

I was able to classify each cluster with a topic on 24/40 of the proposed topics. I think that the human brain has an easier time of deciphering some of the contextual patters when we look at word intrusion – When I was glancing at the output files, I was able to tie 1 or two words together per line that made sense as belonging together, while the rest of the words/bigrams were nonsensical. I believe the number to the left of my proposed topic is the probability that a document, chosen at random, belongs to one of the topics. I think this sum exceeds 1/100% because a document can belong to more than one topic at a time.

**CONCLUSIONS:**

In conclusion, LDA or other forms of topic modeling can help you to generalize main talking points of a collection of documents – Once you have a trained model, you can then apply it to unseen documents and be able to quickly understand what a document is synthesizing, and how it relates to the found/generated probabilities above.

References:

[1] <https://medium.com/@lettier/how-does-lda-work-ill-explain-using-emoji-108abf40fa7d>