Thomas Jarosz
Tjarosz2
669902044

Tech Review of Clustering Algorithms

1. Introduction

For my technology review, I chose to focus on the evaluation and comparison of different (implementations of) algorithms which perform text clustering. I will be evaluating these algorithms in four different categories: execution time, memory usage, F1 score of predicted clusters, and ease of use. Each text clustering algorithm will attempt to cluster a text document into one of three clusters. Each cluster corresponds to a different topic: biology, robotics, and cooking.

2. Data Preparation

The first step to evaluating a set of clustering algorithms is identifying a dataset of text to perform clustering upon. I chose to use a subset of the data provided in a Kaggle post [1]. Specifically, I used the first 10% of the Biology, Cooking, and Robotics datasets (see "biology.csv.zip", "cooking.csv.zip", and "robotics.csv.zip" from the reference website). Each file contains many lines, and each line contains contents of an article pertaining to the title of the csv file. For example, "cooking.csv" contains many lines; each line is the contents of an article pertaining to cooking. I chose to use only the first 10% of these datasets so that these algorithms could be executed in a reasonable amount of time.

In addition to using the datasets provided from Kaggle user "Akshatp", I also utilized some of the code he provided for cleaning up the text data. This cleaning included handling capitalization, html tags, non-alphanumerical characters, whitespace, tokenization, and stop words. I also used the provided code to vectorize my text via Term Frequency – Inverse Document frequency representation. From this cleaned data, I was able to create a dataset of shape 3136 (the number of documents) by 16852 (the size of our text corpus). This would be the final set of data clustered by the algorithms I am evaluating. I also created an array of data containing ground truth classification labels. This array is of length 3136; each element of this array represents the ground truth label for a given document in our dataset which we will perform clustering upon. Ground truth labels of "0", "1", and "2" correspond to documents pertaining to biology, robotics, and cooking respectively.

3. Details on Evaluation Metrics

In this section, I briefly describe details pertaining to my four evaluation metrics.

Execution time: The execution time of a clustering algorithm is measured and reported in seconds. This time includes the time to instantiate and fit a model, along with performing prediction on the provided dataset. If a model only accepts dense matrices, the time to perform conversion of my input data from a sparse matrix to a dense matrix was also included. This

decision was motivated by the fact that Term Frequency – Inverse Document Frequency representations of documents are often sparse. In the "real world" this conversion would be necessary to perform at runtime.

Memory usage: Memory usage of a clustering algorithm is measured and reported in Bytes. This measurement describes the maximum memory used by the Python process to perform clustering. This measurement is a made immediately after performing clustering. This measurement is made using Python's "resource" package, calling "getrusage" and accessing the "ru_maxrss" attribute. To account for the fact that "maxrss" retains the maximum memory used for a given Python process, the Python kernel was refreshed before the evaluation of each clustering algorithm.

F1 score: F1 score is implemented via "Sklearn.metrics.f1_score" [2]. To measure F1 score, "micro" averaging was used. The clusters computed via a given clustering algorithm will be compared to the aforementioned ground truth array when computing the F1 score. One issue that arose during the evaluation of these clustering algorithms is the fact that the name given to a cluster by the clustering algorithm likely will not be the same as the name we assigned in the ground truth label. For example, the ground truth label for all biology documents is "0". A clustering algorithm may instead label all biology documents as "1". The clustering algorithm should not be punished for this. To resolve this issue, some work was necessary to map predicted cluster labels to our ground truth cluster labels. This was done by mapping each ground truth label "L" to the predicted label that occurred most frequently in the set of predictions whose ground truth label is "L".

Ease of use: Unlike the other metrics, ease of use is much more abstract. I will arbitrarily decide the ease of use of a given tool based on several factors including quality of documentation, "bugginess" of the tool, and ability to get the software working quickly.

4. Evaluation of Text Clustering Algorithms

Given the dataset produced as a result of following the procedures described in section 2, I am now able to evaluate different text clustering algorithms. I chose to evaluate the following text clustering algorithms: "sklearn.cluster.SpectralClustering" [2], "sklearn.cluster.MiniBatchKMeans" [2], "sklearn.cluster.Birch" [2], "sklearn.cluster.AgglomerativeClustering" [2], and "sklearn.cluster.KMeans" [2]. See the results of evaluating these clustering algorithms in Table 1 below.

Clustering Algorithm	Execution Time (seconds)	Maximum Memory Usage (Bytes)	F1 Score	Ease of Use (1 = worst, 10 = best)
SpectralClustering	4.396	775,684,096	0.709	**9**
MiniBatchKMeans	**0.116**	**381,128,704**	0.357	**9**
Birch	106.340	3,196,514,304	0.421	**9**
AgglomerativeClustering	99.602	882,409,472	**0.874**	8*
KMeans	0.568	397,291,520	0.302	**9**

Table 1: Evaluation of Text Clustering Algorithms. For each metric, the best performing value is bolded.
* As all clustering algorithms conformed to the Sklearn clustering interface, the usability was roughly the same for all algorithms. AgglomerativeClustering lost one point due to the fact that it does not support clustering upon sparse datasets.

5. Conclusion

Before interpreting the results of my evaluation, it is worth prefacing that there is no "one size fits all" text clustering algorithm. The quality of the performance of a text clustering algorithm is very much dependent upon the size and distribution of your dataset. That being said, the following claims will all be regarding the performance of these algorithms on my specific dataset.

Of the 5 algorithms evaluated, MiniBatchKMeans had the shortest execution time, MiniBatchKMeans utilized the least amount of memory, AgglomerativeClustering had the best F1 score, and all algorithms except AgglomerativeClustering tied for best ease of use. Overall, all algorithms were easy to use and well documented. Personally, I would consider SpectralClustering to be the "overall winner" of my analysis. Although this algorithm does not outperform the others in any single metric, it performed relatively well in all metrics. In my opinion, the metrics with the most significance are F1 score and execution time. It is clear that "SpectralClustering" and "AgglomerativeClustering" outperform the other algorithms in F1 score by a large margin. Though "AgglomerativeClustering" outperforms "SpectralClustering" in this metric, the execution time of "AgglomerativeClustering" is very poor, and it seems that "AgglomerativeClustering" would not scale well for larges sets of text. Given that the ease of use of "SpectralClustering" was just as good as (if not better than) all other algorithms, and its memory usage was not significantly worse than the rest, I would consider "SpectralClustering" to be the best tool to perform clustering on the given dataset.

6. References

[1] Akshatpathak. “Text Data Clustering.” *Kaggle*, Kaggle, 3 Sept. 2018, www.kaggle.com/akshatpathak/text-data-clustering.

[2] Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.