Document Clustering Algorithm using TF-IDF implemented by Heap

Junho Eum

Abstract:

This paper presents an approach to cluster documents using term frequency-inverse document frequency (TF-IDF) to identify the most important terms in each document and then using cosine similarity implemented by heap data structures to sort the documents based on their cosine similarity to the root document.

Introduction:

Document clustering is a crucial task in natural language processing and information retrieval. It involves grouping similar documents based on their content, which can help in many areas such as categorizing news articles, identifying spam emails, and identifying similar legal documents. However, clustering documents can be time-consuming for individuals since it requires human intuition to cluster and store essential documents for companies and big organizations.

In this paper, I present an approach to document clustering that utilizes TF-IDF and cosine similarity implemented with heap data structures. My algorithm aims to guide individuals on how to organize documents inside a directory using heap data structures which has performance optimization since heaps have an average time complexity of O(log n) for insertion and deletion. Clustering algorithms help group similar documents based on their content, which can be achieved by calculating the similarity between each document using cosine similarity and TF-IDF.

Methodology:

The proposed approach uses TF-IDF to identify the most important terms in each document. TF-IDF is a widely used technique in information retrieval to represent the importance of each term in a document. The method assigns a weight to each term in a document based on its frequency in the document and its frequency in the corpus. The approach then uses cosine similarity to calculate the similarity between documents based on the frequency of each term in the document. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space.[1]

First, I conducted a preliminary experimental evaluation of my algorithm using a small dataset. There are a total of 18 documents inside a specified directory filename which function as class labels for the algorithm. Each class label has been selected after reading the document and was labeled in relation to the topic. (*Figure 1*)

Topic	File_name
Machine_Learning	ML_{number}
Biology	Biology
Business	Business_{number}
Ocular_Disease	Ocular_{number}
Music	Music_{number}

Figure 1. Description of document file and file names used to visualize the heap data structure

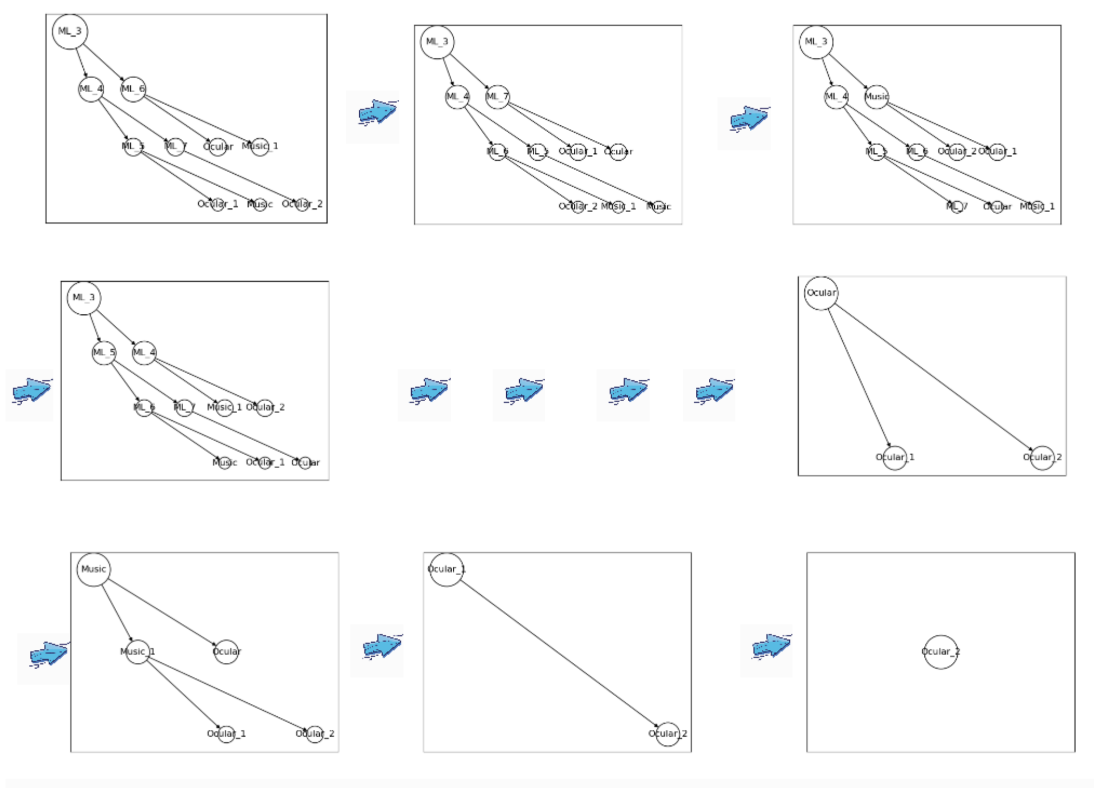
The proposed approach then uses heap data structures to sort documents based on their cosine similarity to the root document. The algorithm begins by initializing a heap data structure with the first document. The heap will generate relevant documents according to the root document. The algorithm loops through each subsequent document, calculates cosine similarity between the new document and each document currently in the heap, and adds the new document to the heap if its cosine similarity is greater than the threshold value. The threshold value determines how similar two documents must be to be considered related. A higher threshold value means that documents must be more similar to be considered related, while a lower threshold value means that documents can be less similar and still be considered related. In

[1] metric - Cosine similarity vs The Levenshtein distance - Data Science ....
https://datascience.stackexchange.com/questions/63325/cosine-similarity-vs-the-levenshtein-distance

the case of heap data structures, a higher threshold value would result in fewer documents being added to the heap data structure, while a lower threshold value would result in more documents being added[2].

The algorithm removes the document with the lowest cosine similarity if the heap is full. The algorithm then repeats the process with the next document, using the updated heap data structure. The algorithm uses networkx and matplotlib libraries to draw a pyramid-like visualization of the heap data structure. (*Figure 2*)

To choose the best heap structure from the iteration, the algorithm calculates the average cosine similarity of the documents in the heap. It keeps track of the heap with the highest average cosine similarity and returns it as the chosen heap. (*Figure 3*)



[2] Zhang, Xiaodan & Hu, Xiaohua & Zhou, Xiaohua. (2008). A comparative evaluation of different link types on enhancing document clustering. 555-562. 10.1145/1390334.1390429.

Figure 2. Iterative visualization of heap data structure of 18 documents with threshold value = 0.01

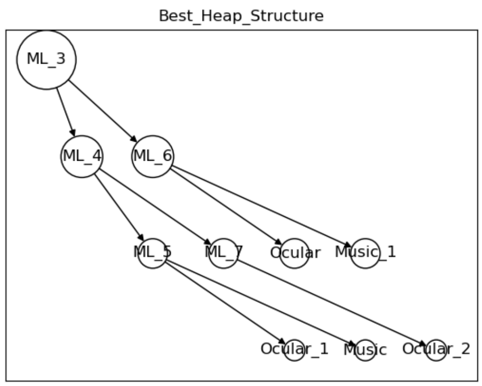


Figure 3. Optimal heap data structure of 18 documents with threshold value = 0.01

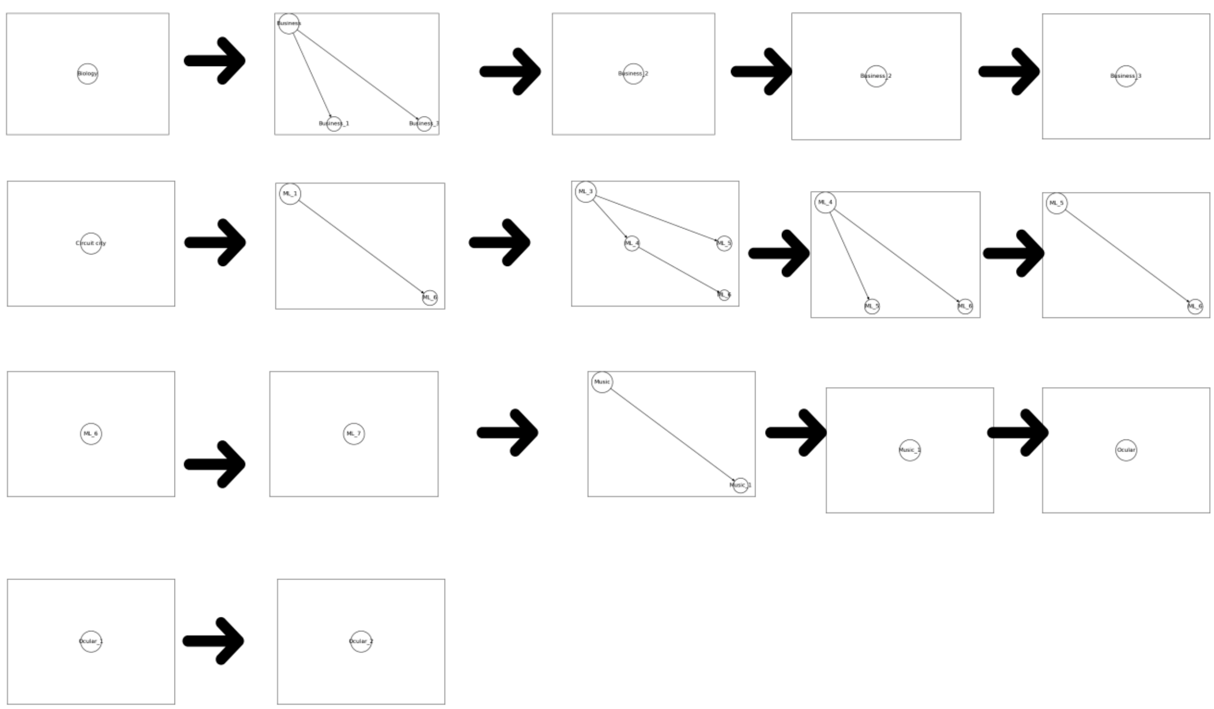


Figure 4. Iterative visualization of heap data structure of 18 documents with threshold value = 0.1

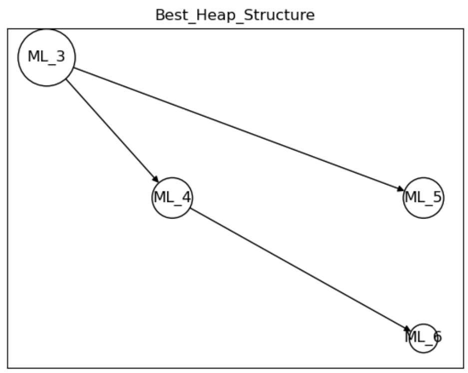


Figure 4. Optimal heap data structure of 18 documents with threshold value = 0.1

Limitation

There are a few limitations to my approach. Firstly, it is difficult to quantify the comparison results in terms of accuracy and computation time between my proposed clustering algorithm and other clustering algorithms, such as K-means. Secondly, there is no perfect solution for finding the optimal threshold for cosine similarity, as it depends on the specific use case and domain structure.

Conclusion

Testing on a small dataset allowed me to explore the algorithm's behavior and assess its feasibility on a limited scale. While the results obtained from this small-scale study cannot be generalized, they provide insights into the algorithm's performance and potential areas for improvement. To assess the generalizability and scalability of our proposed algorithm, I plan to conduct additional experiments on a larger dataset. Specifically, I will compare the performance of our algorithm Latent Semantic Analysis (LSA) clustering algorithm on a dataset comprising 1000 documents, with 200 documents each related to the topics of biology, machine learning, ocular disease, music, and business. I will evaluate the efficiency and effectiveness of the algorithms using several performance metrics, such as clustering accuracy, F1 score, and computational time. By comparing the results of the two algorithms, I aim to demonstrate the superiority of our algorithm in terms of both clustering accuracy and computational efficiency.

I believe that this research will contribute to the development of more effective and efficient algorithms for document clustering, with potential applications in various fields such as information retrieval, text classification, and data mining.

File_Name	Label
'ML_3.docx'	0
'Business_1.docx'	0
ML_5.docx'	0
'Business_3.docx'	0
'ML_7.docx'	0
'Ocular_1.docx'	0
'Ocular_1.docx'	0
'ML_6.docx'	0
'Ocular.docx'	1
'Ocular_2.docx'	1
'ML_4.docx'	1
'Business.docx'	1
'Music_1.docx'	1
'Business_4.docx'	2
'Biology.docx'	2
'ML_2.docx'	3
'Music.docx'	3
'Business_2.docx'	3
'ML_1.docx'	3

Figure 6. Predicted label of each document from K means clustering with elbow method

References

1. metric - Cosine similarity vs The Levenshtein distance - Data Science https://datascience.stackexchange.com/questions/63325/cosine-similarity-vs-the-levenshtein-distance
2. Zhang, Xiaodan & Hu, Xiaohua & Zhou, Xiaohua. (2008). A comparative evaluation of different link types on enhancing document clustering. 555-562. 10.1145/1390334.1390429.