Accelerating Document Retrieval with *k* Means Clustering

Marie Cho

Institution for Computing in Research

July 27, 2022

Abstract

In this work, we propose a method to accelerate k nearest neighbor search with cosine similarity metric by estimating the upper bound of cosine similarity between user queries and object representations. Using trigonometry inequality, the upper bound of cosine similarity between two embedding vectors could be estimated quickly given an appropriate reference point in the embedding space. By analyzing the intuition behind the upper bound estimation in depth, we further found that *k*-means clustering is a practical way of identifying such effective reference points at low cost. When we apply our technique to neural ranking models for top-*k* sentence search, our experimental results on over 300,000 sentences demonstrated that we can accelerate the procedure of selecting the top-5 sentences in terms of relevancy without any approximation over a convention method using full-blown cosine similarity computation.

Keywords: Deep learning, natural language processing, BERT model semantic embedding, cosine similarity

1 Introduction

Document retrieval systems using deep learning have advanced drastically and been a popular and heavily researched topic for the past several years. Latest developments in natural language processing (NLP) allow incorporating contexts and complex lexical structures into document representations or embedding, which in turn enables contextual understanding in the model.

While many modern devices now implement a software to search for file names on the computer, most do not allow searches for the content within files themselves. The nature of the latter is restricted by techniques used with mass files and broad domains. Recent models attempting to successfully achieve this task are based on word frequencies and statistics rather than the semantics of the text.

In this paper, we develop a system incorporating NLP that retrieves semantically similar desktop files to a user query. However, as the field of artificial intelligence (AI) expands, there is a growing need for faster and efficient processing methods. Concerning this, I built a unique approach to comparison techniques to execute faster and more efficient processing.

To retrieve a document is to compare the words of a text file to those of the user's query. The goal of this task is to accurately identify the most similar documents in respect to the requested text. AI and NLP has developed drastically in the past decade and recent efforts to utilize the advancing field of AI and neural networks for text retrieval have yielded promising results. Existing approaches to document retrieval involve statistical techniques and models of word frequency and importance. Although this approach gets the job done, it may be so that the subsequent results is inaccurate or not satisfactory in regards to the user's needs.

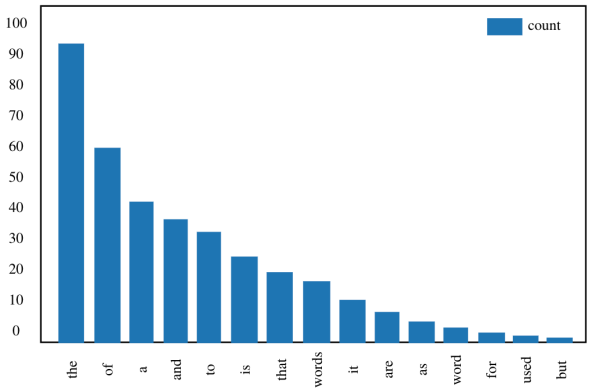

Figure 1: Statistical Word Frequency in a Text

More recent projects that undertook this task went one step further and built programs that achieves the same goal of document retrieval systems, but through a different method. Rather than by implementing statistical algorithms where only numbers are involved, these projects utilize semantic embedding, in which the used model extracts the meanings and connotations of each respective document, allowing for a greater awareness of word relationships and context. Consequently, this correlates to a more precise and well-suited outcome for what the user is searching for. Google and Facebook are some of the prime examples of businesses who apply this algorithm in their company.

Indeed, because both the requests and the contents of documents are expressed in a natural human language, the techniques designed for NLP would provide more appropriate and accurate representations than those used by statistics. However, to feasibly use NLP techniques with a large number of data - the documents - that one would realistically find in the world heavily limits the functions and processes that characterize the docu-

ment retrieval task. In terms of wait time and computing powre, there is a limit to how realistic and worthy implementing this type of software would be. While many devices today contain a software to search for certain files themselves, most do not allow searches for the content within such files. Because content searching is restricted by various factors with consideration to usability and productivity, it may not be reasonable to put this type of search when developing computers.

To addresses these shortcomings, we set out to build a functional program that is capable of identifying and analyzing generated query inputs and desktop text files in a timely, yet accurate manner. Despite the constraints of such NLP techniques within a large-scale environment, there are ways to improve the effectiveness of information retrieval systems by reducing calculations that are not necessarily needed to successfully accomplish the task, with no affect on the end outcome. In this project, we discuss how cosine similarity can be strategically used in a more effective and scalable way.

2 Background

2.1 Artificial Intelligence

Artificial intelligence, otherwise known as AI, refers to systems that perform and improve certain tasks by imitating human intelligence based on previously collected information and data. Artificial intelligence is found everywhere and is continuing to develop today: autonomous cars, facial recognition, personalized recommendations, and surveillance devices are only a few of the many uses of AI in the modern world. Although it is unsurprising that AI grew rapidly since the 1900s, but what is surprising is that humans have contemplated how human intelligence could be artificially mechanized by non-human machines even before 300 BCE - in other words, the idea of AI existed long before the development of technology.

2.2 Natural Language Processing

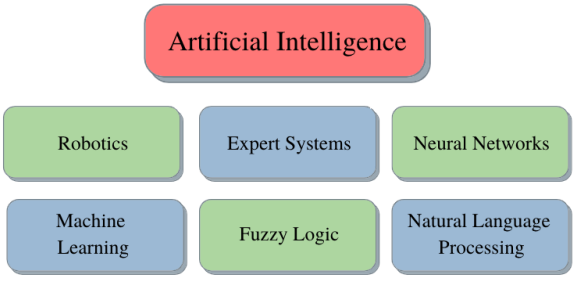

Figure 2: The Branches of Artificial Intelligence

A branch of AI, natural language processing (NLP), is concerned with the lingual interaction between humans and computers and giving computers the ability to understand text and speech in human language. NLP first began after World War II, after realizing the importance of communication despite language barriers and hoped for an automatic translator. In the late 1900s, several divisions of NLP emerged as knowledge and technology advanced, each examining a different aspect of this wide field, such as interchanges between humans and computers as if it were a person to person conversation. Today, NLP is found in translators, spell check, search auto-complete, and virtual assistants.

2.3 The BERT Model

Within the field of NLP, various models and algorithms have been developed to make the devices listed above possible. One such AI language model is the BERT (Bidirectional Encoder Representations from Transformers) Model. BERT is an unsupervised machine learning technique designed by Google in 2018 that allows computers to comprehend the meaning and context of documents. This neural network and transformer-based language processing serves as the bridge between text in human language and embeddings which the computer can read. Using this technique, we are able to make both the text files and user queries comparable in terms of vectors.

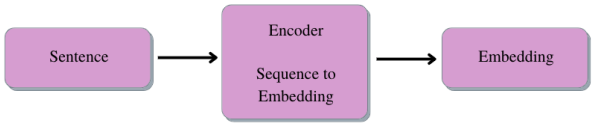

Figure 3: The BERT Model

2.4 Cosine Similarity vs Euclidean Distance

Two important methods of measuring the proximity between vector points in vector space are the cosine similarity and the Euclidean distance. Although each has their own advantage and uses for specific scenarios and the two methods are both widely used, they take different approaches of identifying the closest set of points.

Cosine Similarity is a measure of proximity between two vectors by determining the cosine of the angle between them. Thus, the smaller the angle difference, the higher the similarity score. Two vectors of equal orientation (0°) have a maximum similarity of 1, two vectors that are orthogonal (90°) have a similarity of 0, and two vectors that lie directly opposed to each other (180°) have a minimum similarity of -1. This method is therefore a judgement of orientation and direction, independent of the vectors' magnitude and weight.

$$cos(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

In contrast, Euclidean Distance is the distance between two points in space, or the length of a line segment connecting those two points. The greater the difference between the points, the greater the distance. Thus, this method does consider a vector's

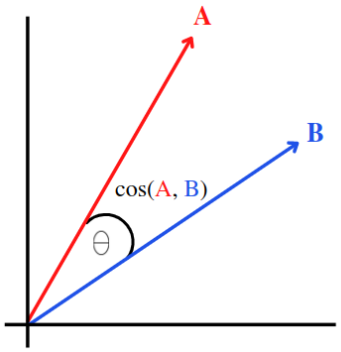

Figure 4: Cosine Similarity

magnitude and weight when calculating the proximity.

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

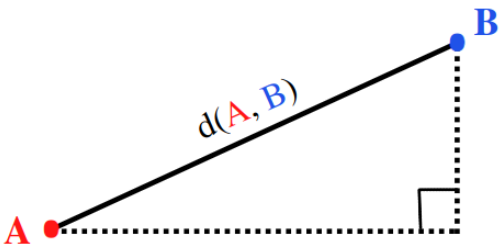

Figure 5: Euclidean Distance

Rather than finding the physical distance between two points, finding the angle measure between vectors is more accurate when measuring document similarity in text analysis. Because texts can vary in terms of size and length yet can contain the same meaning as another, thus the same orientation, cosine similarity is more effective in catching the semantics and therefore more practical in this scenario.

2.5 K-Means Clustering

k-Means Clustering is an unsupervised machine learning algorithm for vector organization by which all the vectors in a space is separated into k groups. The result would be the clusters that new data point belongs to, according the training that the model went through. This form of clustering is often used in content recommendations on streaming sites.

In this process, it will form k random centroids and will assign each vector to the closest centroid, labeled 1, 2, 3 ... k. It uses the distance between points as a measure of similarity, based on the k centroids. Through trial and error, centroids will adjust themselves by taking the average of the vectors in its each respective cluster. This process is repeated over and over until in two consecutive runs, the centroids do not move or the adjustment is minimal.

Ultimately, the centroids which are formed at the end of the algorithm are based on luck and chance. Because the starting positions of the k vectors are completely random, bad starting points could lead to clusterings that are completely wrong. Therefore, another common approach to eliminate this problem is to set the starting k values as some of the data points themselves. Either way, the algorithm will run on the same data set repeatedly until a common solution and clustering formation is found.

Figure 6: K-Means Clustering

3 Data Processing

By importing python libraries PyPDF2 and Spacy, we extracted the string text found in PDF files, detected if they were complete sentences, removed any unnecessary spaces or non-alphabetical characters , and stored the text ready to be inputted into NLP into TXT files. When these texts are inputted into BERT, the model will create a tensor vector with numerical values equivalent to the meaning of the text. Likewise, inputting the string text obtained from the query will result in a numerical tensor vector.

4 Main

Consider two embedding vectors x and y, $\|x\| = \|y\| = 1$.

$$\begin{aligned} \|x - y\|^2 &= \|x - y\|^2 \\ &= x \cdot x^T - 2x \cdot y + y \cdot y^T \\ &= 2 - 2\cos(x, y) \end{aligned} \tag{1}$$

Therefore, finding the most relevant item in terms of cosine similarity is equivalent to finding the one with the least Euclidean distance.

To speed up the top-k nearest neighbor search, it is critical to reduce the number of cosine similarity computations between a query and the items. And, we propose to limit the search to the cluster from k-means clustering. k-means clustering optimizes the overall the squared Euclidean distance in the embedding space between the centroids and the items, and the cosine similarity between a query and a centroid can serve as a proxy to the similarity between a query and the items of the cluster. Hence, we can limit our search to the cluster with the maximum distance between a query and the centroid.

When k=1, it degenerates to a full-blown search. When $k > 1$, it will reduce the overall search time but with the risk of sub-optimal search result. For example, when the embedding vector of a query is on the boundary between two clusters, our acceleration scheme focus on the one with a closer distance. hence leaving out some high-quality items from the search. The larger k for clustering is, the higher the chance of sub-optimal result is.

5 Experimental Setup

In preparation for the experimental runs, we used L2 normalization to make the lengths of all sentence, reference, and query embedding vectors equal to 1 and append their respective file names into a separate array that will be used when displaying the top k search results. Two different modes are used: one without optimization (base) and the other with reference vectors for optimization usingk-means clustering (kmeans) to identify the most effective reference points at a low cost. All loading and query times are recorded for numerical representations of each reference point's efficiency.

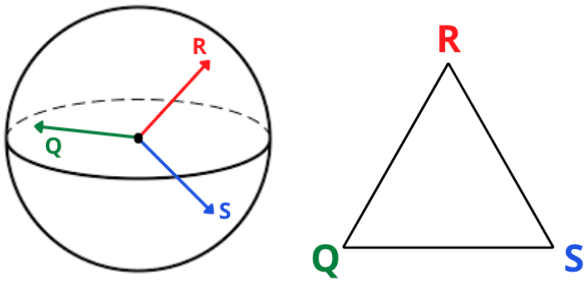

Figure 7: Reference, Query, and Sentence Vectors in Space

Running document retrieval int he base mode means taking the cosine similarity of every sentence with the query and sorting a list with over 300,000 elements greatest to least, and finally retrieving only the first 5 elements. Using this method will result in 100% accuracy, but will take a longer amount of time depending on the scale of data.

For the optimization process, the list of normalized vectors are inputted into k-means clustering with various numbers of centroids (1, 2, 4, 8, 16, 32). From each centroid, 5 random vectors within its respective cluster are stored into a list for later use.

The query vector, Q, is compared with each of the centroids and takes the centroid with the greatest similarity score. The program will then use the correlating list with the 5 random vectors from its cluster as its starting base (instead of [0, 0, 0, 0, 0] for greater optimization and skips) to build the final list with the top 5 nearest neighbors. Finally, for all the sentences in the nearest cluster, it will take the cosine similarity between each said sentence and the query, which then will append to a list and sort, taking only the top 5 - similar to what is seen in the base mode.

When k=1, it will essentially go through the same process as the base mode, as every sentence is included in the one same cluster, and will result in the same outcomes. In contrast, when k>1, the search times will be reduced by kx because we are only examining one out of k clusters, but in exchange for a chance that results may be not 100% accurate as not all possible sentences are included in the similarity process. Despite this, accuracy rates are still very high while exponentially reducing query times.

6 Results and Discussion

6.1 Base

No optimization means taking the cosine similarity of every sentence with the query and sorting a list with over 300,000 elements greatest to least. It took around 17 seconds to load and prepare and about 7 seconds to calculate the top 5 nearest neighbor vectors for each of the 10 queries. While 7 seconds is considerably fast and the base mode is the 100% accurate since it compares the query to every sentence in the text documents, times will get increasingly slower as the system is exposed to more and more data.

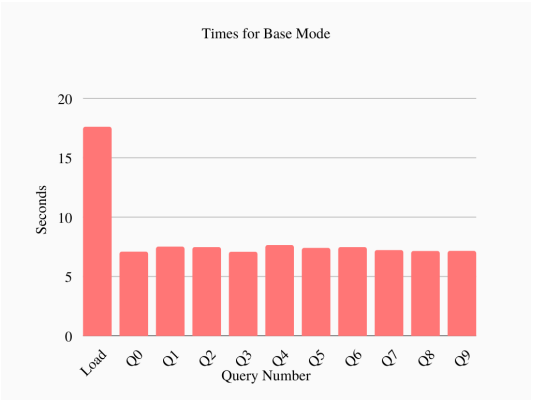

Figure 8: Loading and Query Times for Base Mode

6.2 K-Means Clustering

We experimented 6 runs with the kmeans mode, using 1, 2, 4, 8, 16, and 32 numbers of centroids. As shown in the graph, query times have decreased as the number of centroids increased. Ideally, at some point the time will be essentially 0. However, there is a limit (in this case about 2 seconds), almost as if it was an asymptote, to how fast k-means clustering can be, due to the for loop in which it recognizes if the label of each sentence is part of the closest centroid. Since that step cannot be optimized, there will always be some amount of wait time. Despite this, the overall wait time has reduced significantly.

This comes as a trade-off with loading times and risks of errors, however. As the number of centroids increases, the loading times increase with it. In this case, as k increases from 1 to 32, the loading time increases from 25 seconds to 125 seconds.

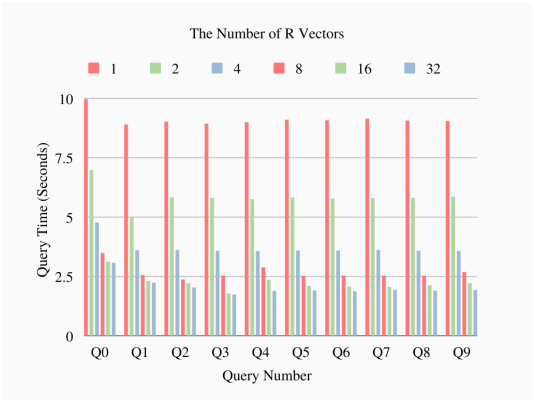

Figure 9: Query Times for K-Means Mode

Errors, because they occur by chance, the accuracy of the search result can waver. As shown in the graph, searches with 1 and 4 clusters have no errors while searches with 16 clusters have a 0.8% error. Ironically, searches with 32 clusters have less errors than those with 8 or 16 clusters. These numbers will vary and fluctuate run by run because it is by chance. Although overall errors increase, the percentage of error are less than 1% .

Note that both factors, the loading time and error percentage, may go above these thresholds as k > 32.

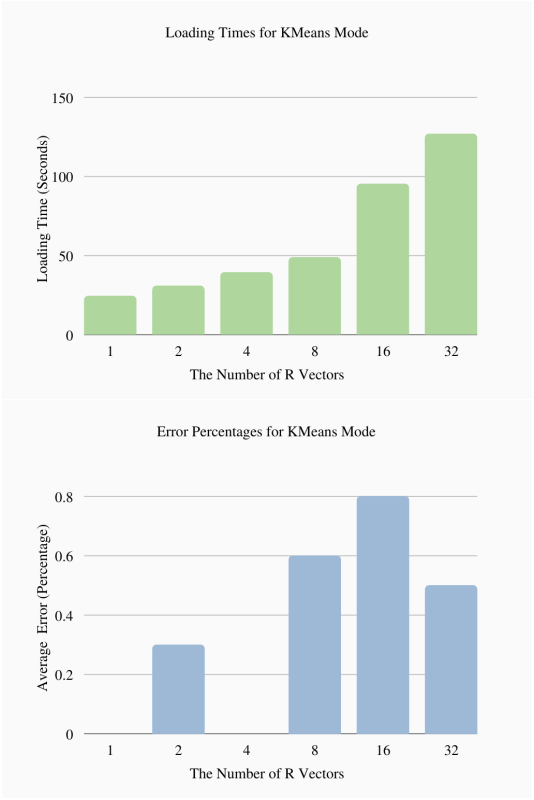

Figure 10: Loading Times and Error Percentages for K-Means Mode

When given a query, the console will return the following of the top 5 near neighbor texts: Order number, similarity score, file name, page number, and the corresponding text. An example of a search result and its query is show below:

Query: Cognitive science is the study of the human mind and brain

Result	Similarity	File	Page	Text
1	0.873885	KrillPlatekGoetzShackelford2007.txt	1	Abstract: Cogni
2	0.842273	MansvelderVerhoogGoriounova2019.txt	5	One of the majc
3	0.838349	Fodor1981.txt	1	Psychological e
4	0.830323	NorenzayanHeine2005.txt	2	Cognitive scien
5	0.83019	Chalmers2011.txt	2	The mathematic

Figure 11: Example of a Search Result and its Query

7 Future Work

Throughout this project, we have been able to produce a data retrieval system through the use of natural language processing in combination with various algorithms, such as cosine similarity and k-means clustering. While our program has been optimized and improved, we believe that further steps could be taken in the future to make this project more advanced. These are multiple next steps that can be taken to build on these findings:

1. Implementing the ability to search and sort through a even larger mass of files
2. Improving the accuracy of search results
3. Optimizing the loading times

Acknowledgements

We would like to sincerely thank: Optiver for providing us their space during this internship, Mark Galassi and Rhonda Crespo for running the ICR and giving us this research opportunity, our fellow interns for being supportive along the way, and finally our mentor Jim Davies for guiding us through this project.

References

[1] Rebecca Reynoso (2021) *A Complete History of Artificial Intelligence*, G2.

[2] Shamala Gallagher, Anna Rafferty, and Amy Wu (2004) *Natural Language Processing*, Stanford University.

[3] Rani Horev (2018) *BERT Explained: State of the art language model for NLP*, Towards Data Science.

[4] Daniela M. Witten and Robert Tibshiran (2010) *A Framework for Feature Selection in Clustering*, Journal of the American Statistical Association.

[5] Nandan Thakur, Nils Reimers, Johannes Daxenberger, and Iryna Gurevych (2021) *Augmented SBERT: Data Augmentation Method for Improving Bi-Encoders for Pairwise Sentence Scoring Tasks*, Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics.