Approximate Nearest Neighbor based information extraction with text and image data

Malik Naik Mohammed, Yves Junior Kwame Kamgaing, Sai Charan Gandi, Madhusudhan Rao Atmakuri
Kennesaw State University
1100 South Marietta Pkwy SE
Marietta, Georgia 30060-6221, USA
{mmoham25, ykwameka, sgandi, matmakur}@students.kennesaw.edu

***Abstract*—Most of the search engines use algorithms like Best Match 25 [1] that perform well and return the top-ranked results, but they lack the ability to understand the semantics of the user that they are searching for. The main task of this project is to apply the various deep learning techniques to build the search engine that gives the most relevant results using the search query using the pre-trained models. Our main objective is to use deep learning to rank highly similar results at scale. This project also deals with the image data using the Image Captioning model that was trained on the Open Images V6 dataset [2]. We have successfully vectorized the text data from the 200,000+ Jeopardy! Questions dataset [3] and wrote the search engine to search for the given query by vectorizing the search query and fetches the results with the least cosine distance.**

***Index Terms*—Approximate Nearest Neighbor Search, Vector Modeling, Deep Learning, Transfer Learning, Microsoft SPTAG**

I. Introduction

Deep Learning Information Retrieval (IR) is a booming area of research. Research in this field focuses on retrieving the most relevant search results based on the meaning of the search result, not just the keywords. The state-of-the-art technologies generally involve taking deep neural networks (such as Universal Sentence Encoder, Google's BERT, and many more) and training them to rank search results. However, some search engines make use of text matching algorithms like Best Match 25 [1]. These algorithms work by taking the term frequency and other word patterns into account and work surprisingly well.

Nearest neighbor search (NNS) picks up an item from the database that has the smallest distance to the given query. It is a key and fundamental activity in many applications in various domains, including machine vision, databases, multimedia, machine learning and recommendation systems. [4]. Despite much research in this domain it is generally very expensive to find the nearest neighbor in the high dimensional euclidean or cosine space because of the *curse of dimensionality* [5]. Tests demonstrated the way that precise techniques can seldom beat the animal power straight output strategy when dimensionality is high [2] (e.g., more than 20). All things considered, returning adequately close by objects,

Various experiments have shown that exact methods can rarely outperform the brute-force linear scan method when dimensionality of the search space is high [6]. Nonetheless, returning approximate nearest neighbor search (ANNS) that are the closest neighbors of the given text, are useful for many practical problems like search engines and can be done efficiently, thereby drawing in huge research contributions.

The most common problem when it comes to Machine Learning is over-fitting. Even if the model gives better search results for programming-related questions, that doesn't mean that it'll work for searching the research articles. Generally, machine learning-based models are often slow and unscalable. In this project, we tackle all these problems using the deep learning networks and approximate nearest neighbor search to implement the search engine that accurately ranks the search results based on the given query with a rapid and scalable solution. Microsoft is shedding its proprietary software maker image and getting closer to the open-source community. Microsoft made its powerful algorithm used in Bing's core search engine open-source [7]. This work uses the fast approximate nearest neighbor search library by Microsoft called SPTAG[1].

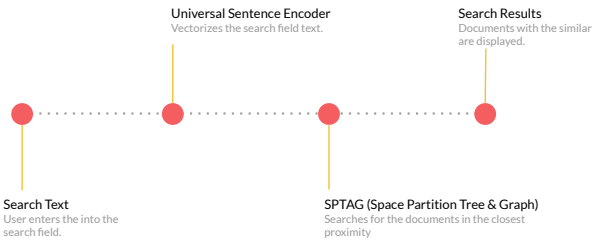

Fig. 1: Search Workflow.

II. Related Work

W. Li et al. [4] classified the state-of-the-art ANNS algorithms into three main categories: Partition-based, Graph-based, and Hashing-based. Partition-based methods divides the high dimensional space into multiple disjoint regions. Graph-based methods build a proximity graph where each data

[1]SPTAG - Space Partition Tree And Graph, a library for fast approximate nearest neighbor search

point represents a node and edges represents the neighbor-relationship. The main idea of these methods is a neighbor's neighbor is likely to also be a neighbor. [4]

Hashing-based algorithms transform data point to a low-dimensional representation, so each point could be represented by a short code or also referred to as hash code. There are two main sub-categories in this class: locality sensitive hashing (LSH) and learning to Hash (L2H). The LSH methods rely on a family of locality sensitive hash functions that map similar input data points to the same hash codes with higher probability than dissimilar points. Learning to Hash methods fully make use of the data distribution to generate specific hash functions, leading to higher efficiency at the cost of relinquishing the theoretical guarantees. [4]

kd-tree [8] has long been deemed unsuitable for the nearest neighbor search in high dimensional space. The performance of kd-tree do not show significant improvements over brute-force nearest-neighbor search in medium to high dimension data. kd-tree are successfully used for approximate search. Ram et al. [9] proposed the kd-tree based approximate search schemes with $O(d\ log\ d + log\ n)$ query time for datasets with $n \times d$ dimension with the increase in the search accuracy.

The k-means tree (KMT) uses oblique trees that creates optimal oblique clustering trees [10]. Oblique decision trees use oblique decision boundaries to simplify the boundary structure. The computational complexity of the k-means tree is linear over number of samples.

The Relative Neighborhood Graph (RNG)[2] is an undirected graph with a set of points in the Euclidean plane by connecting two points p and q by an edge, where there is a thrid point z that is closer to both p and q than they are to each other. [11]

There has been remarkable research in text and image retrieval systems over the last few years. Krizhevsky et al. [12] trained a CNN to classify the ImageNet dataset consisting of over 1 million images into 1000 different classes. As a result, the need for building search engines to retrieve the data from the IoT devices has grown. Kaushik et al. [13] used a Natural Language Processing (NLP) based approach to summarize the search query and perform the search to display the relevant results along with the ranking of the results. We employ the Image Captioning and Approximate Nearest Neighbor search to rank the similar results with the closest proximity. Microsoft's SPTAG provides two methods: kd-tree and relative neighborhood graph (SPTAG-KDT) and balanced k-means tree and relative neighborhood graph (SPTAG-BKT). The SPTAG-KDT is helpful in index building cost, whereas SPTAG-BKT is helpful in terms of search accuracy in very high-dimensional data. To the best of our knowledge, at the time of this paper, there is no such implementation or work that has been done so far that uses Microsoft's SPTAG and Image Captioning at the same time to perform the nearest neighbor search.

[2]RNG - Relative Neighborhood Graph

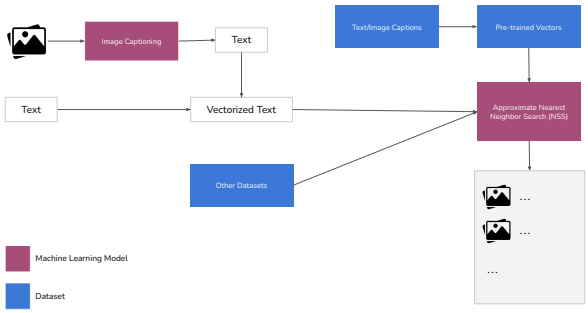

Fig. 2: Proposed Architecture

III. Methodology

We propose the use of Space Partition Tree And Graph (SPTAG) [14] library, which is built by Microsoft and currently used in Bing's core search engine [7]. The proposed architecture of the model workflow is shown in Fig. 2. This library is for a large-scale vector approximate nearest neighbor search scenarios released by Microsoft Research (MSR) and Microsoft Bing. It uses the vector data to perform the nearest neighbor search. The SPTAG works with the two basic modules: index builder and searcher. The Relative Neighborhood Graph (RNG) is built on the k-nearest neighborhood graph [15], [16] for enhancing the connectivity. Whereas, the Balanced k-means trees are used to replace kd-trees to avoid the inaccurate distance bound estimation in kd-trees for relatively very high-dimensional vectors. The search is first started in the space partition trees for finding several seeds to start the search in the RNG. The searches in the trees and the graph are iteratively done.

The image captioning model is built with the Encoder-Decoder architecture. The encoder network which generates the low resolution representations of the given input image. We used the ResNet-152 as the encoder network architecture by replacing the last fully connected layer of ResNet-152 with the linear layer with the 2048 input features and 256 output features. The encoder has a total of 60,192,808 trainable parameters. The decoder network is responsible for generating the image captions and it is trained on the captions of the image. The decoder network consists of embedding layer, long short-term memory (LSTM) layer, and a linear layer. The embedding layer is mostly used to store the word embeddings and retrieve them with indices. Generally, the input to the embedding layer is an array of indices, and the output is their corresponding word embeddings. The embedding layer has the 16,577 number of embeddings i.e; size of the dictionary of embeddings. The size of the embedding vector is 256. The long short-term memory (LSTM) layer 256 number of number of expected features in the input from the embedding layer. There are 512 number of features in the hidden state. The LSTM layer has only one recurrent layer and there is no stacking of LSTM layers in our architecture. Finally, we have the linear layer as the output layer in the decoder network with

Column Name	Column Description
Question	the question text.
Answer	the actual answer.
Category	The category of the question, e.g. ”History”.
Air_Date	the show air date in YYYY-MM-DD format.
Round	It could be one of ”Jeopardy!”, ”Double Jeopardy!”, ”Final Jeopardy!” or ”Tiebreaker”.
Value	The value of the question.
Show_Number	The show number.

TABLE I: 200,000+ Jeopardy! Questions column descriptions

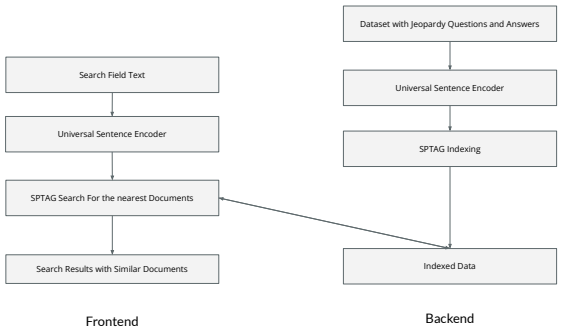

Fig. 3: High-level Architecture

the 512 number of input features and 16,577 number of output features. The maximum sequence length of this network is 80 i.e; the image caption is maximum of 80 words.

D. Cer et al. [17] proposed the Universal Sentence Encoder model for encoding sentences into embedding vectors that particularly target transfer learning to various other NLP tasks. The models proposed in [17] are efficient and result in accurate performance on varied diverse transfer tasks. The two encoding models proposed are Transformer-based and Deep Averaging Network (DAN). These two variants of the encoding models allow for trade-offs between accuracy and computing resources. We used the Universal Sentence Encoder [17], a pre-trained model, to generate the vector representation of the text data and then start indexing the vector data and perform furthur search queries on the . The SPTAG supports L2 and Cosine distances on the vector data. Fig. 1 shows the workflow of the search query in our implementation.

We combined the image captioning model and approximate nearest neighbor search models into one architecture and deployed the model on the server with the exported trained model state dictionaries. The web interface was implemented using the Flask[3] that takes the search queries and returns the results with the closest similarity. Our high-level architecture with each various components is shown in Fig. 3.

IV. Experimental Evaluations

This section briefly discusses the experimental setup, metrics, datasets, and results of our model architecture.

A. Experimental setup and Metrics

The experimental results were performed on the Lambda Vector with Ubuntu 20.04 LTS (Focal Fossa) on an Intel Core i9-10920X processor with 24 CPU cores (4.6 GHz frequency). The hardware also had 64 GB RAM and two NVIDIA GeForce RTX 2080 Ti graphics cards. Our primary programming language is Python version 3.9. Since we are dealing with the vector data and the evaluation metric is Cosine distance or L2 distance. The smaller the distance between the search query and the top result, the better, as a smaller distance means high similarity. Also, note that the Cosine distance is different from the Cosine similarity.

B. Dataset descriptions

For this work we used two different datasets. One dataset is text-based and the other dataset is image-based with captions. The first dataset that we used for searching text and indexed the dataset is the 200,000+ Jeopardy! Questions [3] from Kaggle, which contain the jeopardy questions and answers. The main reason we choose this dataset is because we were looking for a dataset with different types of data that we could perform search with different types of search queries. We found this dataset that met all our requirements. The dataset contains seven columns and 2,16,930 samples. The description about the columns is described in Table. I.

For this work we are interested in the *Category*, *Question* and *Answer* columns. We preprocessed the data into new dataset reducing the dimension of the original data and cleaning the *Question*, and *Answer* by removing the stopwords, punctuations, lowercasing the text, etc. As discussed before we used the Open Images V6 dataset [2] for the image captioning model. It is the largest annotated image dataset that is used in training the latest deep convolutional neural networks for

[3]Flask - a micro web framework in Python

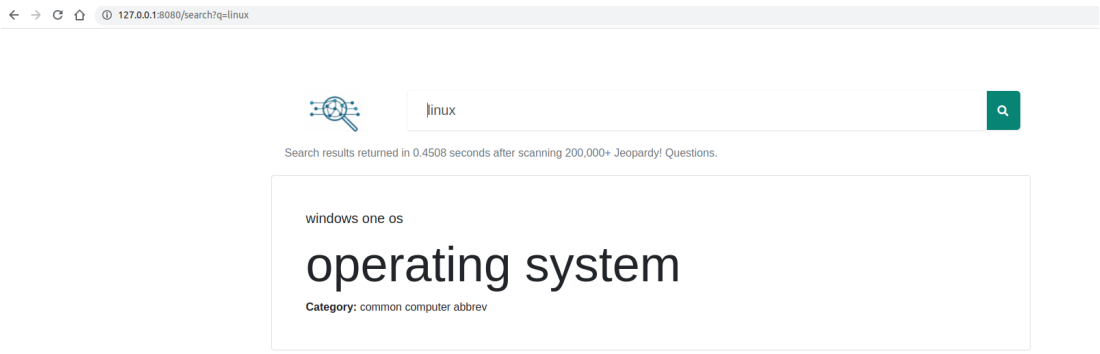

Fig. 4: Search result

machine vision tasks. The Open Images V5 dataset includes 9M images annotated with 36M image-level labels, 15.8M bounding boxes, 2.8M instance segmentations, and 391k visual relationships [18].

The Open Images V6 [2], expands the annotation of the Open Images V5 dataset with a large set of new visual relationships, human action annotations, and image-level labels [18]. It also adds the localized narratives [19], a completely new form of multimodal annotations.

C. Results

We generated the word embedding of all the questions and answers from the text dataset into the real-valued vectors real-valued vector that encodes the meaning of the word using the Universal Sentence Encoder from the Tensorflow Hub as the embedding tool. Each vector is of 1×512 dimension. The total dimension of the data that was generated was 216930×512. After generating the word embeddings we transformed it into the format accepted by the Microsoft's SPTAG library for indexing. The accepted format is as follows:

$$metadata\backslash t < v_1 > | < v_2 > |...| < v_{512} > |$$

The metadata can be anything that is returned when a match to the vector in that row is matched by the indexer. v_1, v_2, ..., v_{512} are the real values at 1, 2, ..., 512 index of the word embedding column vector.

Finally, we indexed the vectored data using the Microsoft's SPTAG using the Balanced k-means Tree and relative neighborhood graph (SPTAG-BKT) as it has good search accuracy with very high-dimensional data. When given the search query like "Linux" the model was able to rank the results with "Operating Systems", "Windows", etc. even though the dataset doesn't contain the word "Linux" in it. The response of the results was within 0.5 seconds. The executed result can be seen in the Fig. 4.

V. Conclusion and Future Work

We successfully preprocessed, trained, and deployed the model onto the server. The indexer was built using the Balanced k-means Tree and relative neighborhood graph (SPTAG-BKT) algorithm. Then, we trained the image captioning model to return captions (e.g., text) from the image and then feed it to the vectorizer that generates the word embedding using Universal Sentence Encoder as shown in the architecture in Fig. 2. For the future work, we can try indexing the text corpus using the kd-tree and relative neighborhood graph (SPTAG-KDT) and compare it's performance.

References

[1] Wikipedia, "Okapi BM25 — Wikipedia, the free encyclopedia," https://en.wikipedia.org/wiki/Okapi_BM25, 2022, [Online; accessed 15-April-2022].

[2] "Open images dataset v6 + extensions." [Online]. Available: https://storage.googleapis.com/openimages/web/index.html

[3] B. Tunguz, "Towards efficient and intelligent internet of things search engine," *Kaggle.com*, vol. 9, pp. 15 778–15 795, 2021.

[4] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1475–1488, 2020.

[5] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, ser. STOC '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 604–613. [Online]. Available: https://doi.org/10.1145/276698.276876

[6] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the 24rd International Conference on Very Large Data Bases*, ser. VLDB '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, p. 194–205.

[7] C. Waldburger, "'how tall is the tower in paris?' how bing knows its about the eiffel tower," Dec 2019. [Online]. Available: https://blogs.microsoft.com/ai/bing-vector-search/

[8] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, p. 209–226, sep 1977. [Online]. Available: https://doi.org/10.1145/355744.355745

[9] P. Ram and K. Sinha, "Revisiting kd-tree for nearest neighbor search," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1378–1388. [Online]. Available: https://doi.org/10.1145/3292500.3330875

[10] D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown, "Hhcart: An oblique decision tree," 2015. [Online]. Available: https://arxiv.org/abs/1504.03415

[11] K. J. Supowit, "The relative neighborhood graph, with an application to minimum spanning trees," *Journal of the ACM*, vol. 30, no. 3, pp. 428–448, Jul. 1983. [Online]. Available: https://doi.org/10.1145/2402.322386

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[13] K. Sekaran, P. Chandana, J. R. V. Jeny, M. N. Meqdad, and S. Kadry, "Design of optimal search engine using text summarization through artificial intelligence techniques," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, p. 1268, Jun. 2020. [Online]. Available: https://doi.org/10.12928/telkomnika.v18i3.14028

[14] Q. Chen, B. Zhao, H. Wang, M. Li, C. Liu, Z. Li, M. Yang, and J. Wang, "Spann: Highly-efficient billion-scale approximate nearest neighbor search," in *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*, 2021.

[15] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li, "Scalable k-nn graph construction for visual descriptors," in *CVPR 2012*, 2012, pp. 1106–1113.

[16] J. Wang, N. Wang, Y. Jia, J. Li, G. Zeng, H. Zha, and X. Hua, "Trinary-projection trees for approximate nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 388–403, 2014.

[17] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," *CoRR*, vol. abs/1803.11175, 2018. [Online]. Available: http://arxiv.org/abs/1803.11175

[18] "Open images v6 - now featuring localized narratives," Feb 2020. [Online]. Available: https://ai.googleblog.com/2020/02/open-images-v6-now-featuring-localized.html

[19] J. Pont-Tuset, J. R. R. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari, "Connecting vision and language with localized narratives," *CoRR*, vol. abs/1912.03098, 2019. [Online]. Available: http://arxiv.org/abs/1912.03098