Data Retrieval via Natural Language Processing

Marie Cho, Changxu Liu

Insitute for Computing in Research

August 4th, 2022

Abstract. Modern devices allow users to search for certain files, but solely utilize the names of the files, instead of the contents stored within. As humanity progresses technologically, the increasing amounts of data will serve as a barrier to a practical application of the latter task in terms of time and cost. This will necessitate accurate methods for finding information in a timely and effective manner. In this work, we create a system to search for relevant content within texts and images in respect to a user's queries utilizing NLP that is both productive and accurate when tested on over 300,000 sources of data.

Keywords: Deep learning, natural language processing, neural networks

1 Introduction

Data 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 data representations or embeddings, which in turn enables contextual understanding in the model.

While many modern devices now implement 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.

2 Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is concerned with the ability for computers to understand human speech and text. The field of NLP began with hopes for an automatic translator after realizing the importance of interlingual communication. Today, NLP is used in translators, grammar and spell check, and virtual assistants.

2.2 Neural Networks

2.2.1 Recurrent Neural Networks

Recurrent neural networks are commonly used in tasks concerning sequential data, such as that which is found in natural language processing. These are neural networks that contain a short term memory by being able to factor in the result of a previous element of a sequence into the processing of the next.

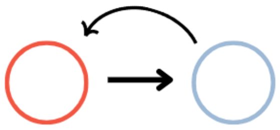


Figure 1: Recurrent Neural Networks

2.2.2 Convolutional Neural Networks

When dealing with visual data, convolutional neural networks are frequently used to extract their features. Convolutional neural networks function by passing filters, called kernels, through an input. These kernels represent various features that may be present in the data, such as an edge or basic geometric shape. By performing matrix multiplication, activation values are created to represent how well different sections of an image match up with the kernel, and the feature it represents. The results may be passed through multiple layers to find increasingly sophisticated features.

Figure 2: Convolutional Neural Networks

2.2.3 Computer Vision

Computer vision (CV) is concerned with the ability for computers to understand and process visual data. The field of computer vision started with the detection of basic edges and geometric shapes. Applications of this technology include self driving vehicles, facial recognition, and optical character recognition.

2.3 Transformers and the BERT Model

A transformer model is a neural network that adopts self-attention, the ability to weigh the significance of a part of data. Transformers are primarily used in NLP and CV. One transformer model highlighted in this project is the Bidirectional Encoder Representations from Transformers (BERT) Model, an unsupervised machine learning algorithm by Hugging Face to allow computers to understand the meanings of specific text and convert human language into semantic embeddings, or vectors.

2.4 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. In this process, it will form k random centroids and will assign each vector to the closest centroid, labeled 1, 2, 3 ... k. 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 multiple iterations until in two consecutive runs, the centroids do not move or the adjustment is minimal.

3 Data Processing

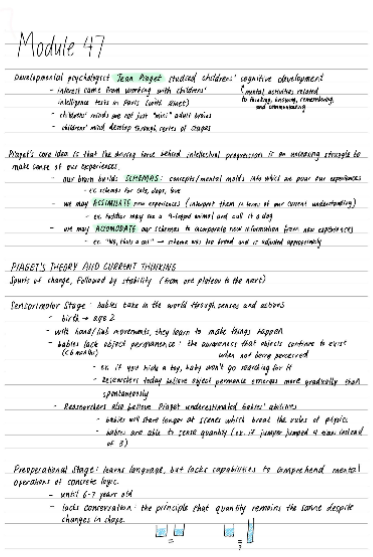
3.1 PDF Text Extraction

For most PDFs, the PyPDF2 Python module was used to extract printed text data.

However, in other cases, PDF files may be scanned documents, in which there will only be image data present in the document, which is not searchable. In order to extract the text data within, optical character recognition (OCR) is used.

Initially, we used the Pytesseract OCR module to effectively pull printed text data out of an image. However, it was unable to properly read handwritten text data accurately with the vast amount of variation present.

A separate system was implemented to convert handwritten text into a retrievable format. To be able to properly extract handwritten text data, the handwritten text must first be located and isolated within each page before being fed into the model. Our method involved utilizing transforms in the OpenCV library to highlight text contours in scanned data.



Module 47

Developmental psychologist Jean Piaget studied childrens' cognitive development
- interest came from working with childrens' intelligence tests in Paris (with Binet)
- childrens' minds are not just "mini" adult brains
- childrens' mind develop through series of stages

(mental activities related to thinking, knowing, remembering, and communicating

Piaget's core idea is that the driving force behind intellectual progression is an unceasing struggle to make sense of our experiences.
- our brain builds SCHEMAS: concepts/mental molds into which we pour our experiences
  - ex. schemas for cats, dogs, love
- we may ASSIMILATE new experiences (interpret them in terms of our current understanding)
  - ex. toddler may see a 4-legged animal and call it a dog
- we may ACCOMODATE our schemas to incorporate new information from new experiences
  - ex. "No, that's a cat" → schema was too broad and is adjusted appropriately

PIAGET'S THEORY AND CURRENT THINKING
Spurts of change, followed by stability (from one plateau to the next)

Sensorimotor Stage: babies take in the world through senses and actions
- birth → age 2
- with hand/limb movements, they learn to make things happen
- babies lack object permanence: the awareness that objects continue to exist when not being perceived (<6 months)
  - ex. if you hide a toy, baby won't go searching for it
  - Researchers today believe object permanence emerges more gradually than spontaneously
- Researchers also believe Piaget underestimated babies' abilities
  - babies will stare longer at scenes which break the rules of physics
  - babies are able to sense quantity (ex. if jumper jumped 4 times instead of 3)

Preoperational Stage: learns language, but lacks capabilities to comprehend mental operations of concrete logic.
- until 6-7 years old
- lacks conservation: the principle that quantity remains the same despite changes in shape.

=
=
?

Figure 3: Handwriting document to preform OCR on.

1. The scanned image is converted to grayscale and its colors are inverted.
2. A threshold is applied to convert each pixel to either 0 or its max value. This helps improve the contrast, makes the lighting consistent, and reduces any noise.
3. The remaining contours on the image are dilated in the x direction to consolidate each line of text into one shape.
4. The coordinates of a rectangular bounding box are recorded and used to crop out each individual line.

This method presents limitations to the format of image data in which text may be detectable. However since our main focus is

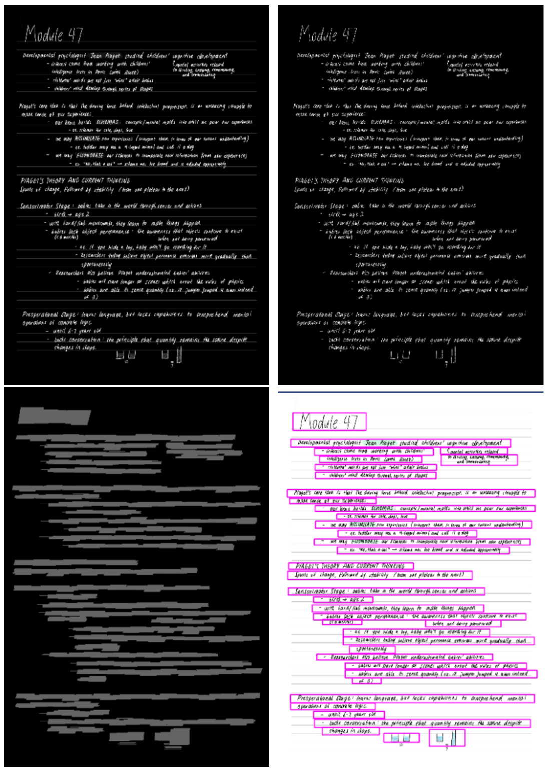


Figure 4: OCR

on scanned documents for data retrieval, this should be adequate. Then, with each line of text cropped, it can be fed into a convolutional neural network to perform feature extraction. Afterwards, the resulting feature map can be put into a recurrent neural network to propagate relevant features. Finally, we can use Connectionist Temporal Classification (CTC) to decode our data into text.

3.2 Image Captioning

For pure images and photographs that don't contain any text, generating a caption to describe the image will need to be done to make it searchable. To do this, a convolutional neural network is utilized to extract features from the input image. We used the pre-trained 50 layer variant of the deep ResNet model. Unique for its residuals which pass data through certain layers, it helped reduce the degradation problem, where the accuracy of neural networks decreased with a greater number of layers.

Because classification isn't needed, the final fully-connected layer is removed from the network, and the features are then passed into a recurrent neural network, which decodes the feature vector into a basic caption that can be used to describe the image.

This model was trained on the Flickr8K dataset, which we believed would represent photos an individual may need to search through on a computer.

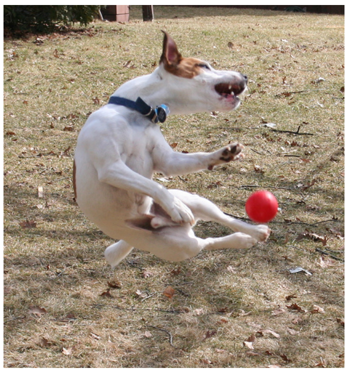


Figure 5: "a white dog is jumping up to catch a ball in a field."

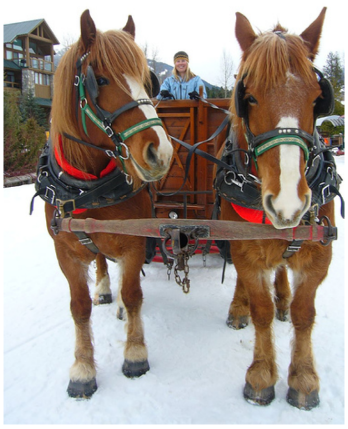


Figure 6: "two horses pull a carriage driven by a man in a blue jacket. "

4 Data Retrieval

4.1 The BERT Model

Within the vast field of NLP are various language models that have been developed to make such innovations possible. The BERT Model, an unsupervised machine learning algorithm, allows computers to understand the meanings of specific text. This serves as the bridge between human language and semantic embeddings, allowing the text files and queries to be comparable numerically. When a text is inputted into BERT, the model will create a tensor vector with numerical values equivalent to the meaning of the text.

4.2 Determining Proximity

Two important methods of measuring the similarity between vectors in space are the Cosine Similarity and the Euclidean Distance. Although each has their own advantages, they take different approaches of determining proximity.

4.2.1 Cosine Similarity

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.

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

4.2.2 Euclidean Distance

In contrast, Euclidean Distance is the distance between two points in space, or the length of a line segment connecting those two points. Thus, this method does consider a vector's magnitude and weight when calculating the proximity.

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

Rather than finding the physical distance between two points, finding the angle measure between vectors is more accurate when using NLP. 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.

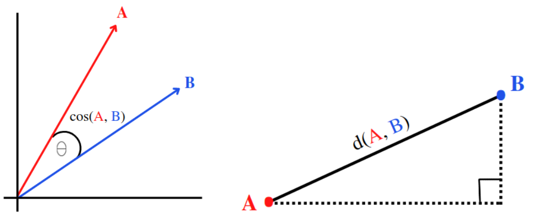


Figure 7: Cosine Similarity vs Euclidean Distance

4.3 Acceleration through K-Means Clustering

In preparation for the experiment, we used normalized the vectors using L2 normalization. Two different modes are used: base (without optimization) and kmeans (with optimization using k-means clustering and reference vectors). All loading and query times are recorded for numerical representations of each reference point's efficiency.

Finding the most relevant item in terms of cosine similarity is equivalent to finding the one with the lowest 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. Hence, we can limit our search to the cluster with the minimum 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 results. For example, when the embedding vector of a query is on the boundary between two clusters, our acceleration scheme focuses on the one with the closer distance, hence leaving out some high-quality items from the search. The larger the k for clustering, the higher the chance of sub-optimal results but the lower the wait time.

5 Results

5.1 Base

The base mode 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 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.

5.2 K-Means Clustering

We experimented with 6 runs using various k values in the kmeans mode. 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 seconds. However, there is a constant factor present that cannot be optimized, so there will always be some amount of wait time. Despite this, the

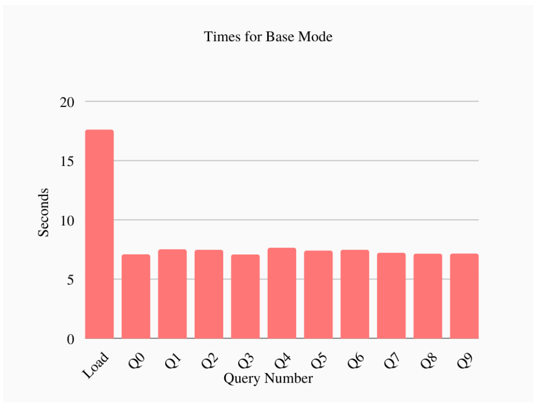


Figure 8: Loading and Query Times for Base Mode

overall wait time has reduced significantly.

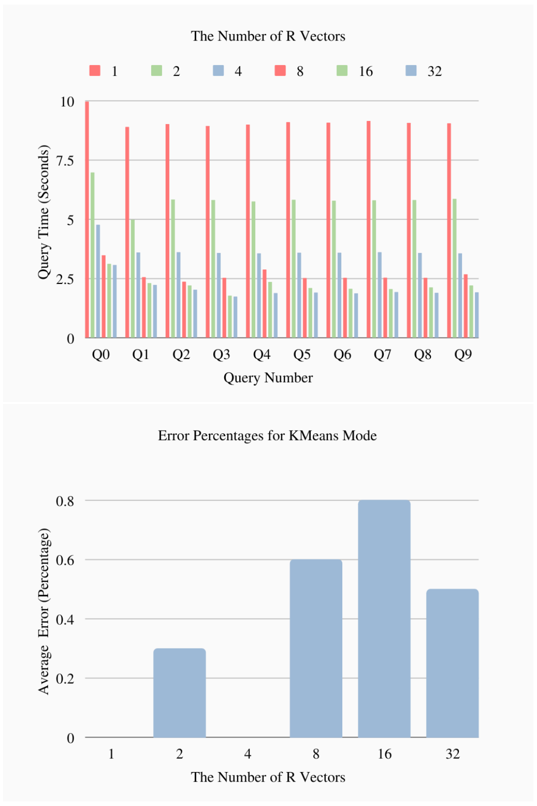


Figure 9: Query Times and Error Percentages for K-Means Mode

This comes as a trade-off with a risk of errors and inaccuracy. However, because errors occur by chance, the accuracy of the search result can waver. Although overall errors increase, the average percentages are less than 1%. Note that both factors may go above these thresholds as k increases above 32.

6 Future Work

Throughout this project, we have been able to produce a data retrieval system through the use of natural language processing in addition to machine learning and neural networks. 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. Optimizing memory usage for the ability to search and sort through a even larger mass of files
2. Improving the accuracy of search results
3. Generating more detailed image captions
4. Better format recognition in OCR

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.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv

[7] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay (2011) *Scikit-learn: Machine Learning in Python*, JMLR

[8] Paszke et al (2019) *PyTorch: An Imperative Style, High-Performance Deep Learning Libary*, Curran Associates Inc.

[9] C.R. Harris, K.J. Millman, S.J. et al. van der Walt (2020) *Array programming with NumPy*, Nature

[10] Kaiming He, Xiangyu Zhuang, Shaoqing Ren, Jian Sun (2015) *Deep Residual Learning for Image Recognition*, arXiv

[11] Micah Hodosh, Peter Young, Julia Hockenmaier (2013) *Flickr8K Dataset*, Journal of Artificial Intelligence Research

[12] Ray Smith (2007) *An Overview of the Tesseract OCR Engine*, IEEE

[13] Ralf C. Staudemeyer (2019) *Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks*, arXiv

[14] Bradski, G. (2000) *The OpenCV Library*, Dr. Dobb's Journal of Software Tools