Investigating the Overlap and Divergence in Search Results between Annoy and Lucene

Bhavya Sharma, Urvika Gola, Elliot Justice
CSC 583 Project Report

Table of Contents

Abstract

Neural networks are often shown to produce better results for information searches compared to traditional retrieval models like Lucene. We note that these networks are often trained on the same data sets used for the results analysis. Although this is standard practice for machine learning, the approach may end up leading to the models overfitting on the data sets and biasing the analysis. In this project, we compare/contrast the performance of a transformer model against Lucene in a domain specific search task.

Data Set and Preprocessing

We selected a JSON dataset containing the metadata and abstracts of 2.2 million papers from Arxive.org [2]. The idea behind this choice was that the large amount of domain specific terminology would perhaps insulate us from potential overfitting on data sets used to train any transformer model we selected and eyeball how well the model was generalizing. Initially, we intended to index and search on all fields in the data set, but time and computational constraints limited us using the title and abstract of each entry in the data set. We note that the data set contained entries for retracted papers, which are identifiable by the abstract replaced with a note indicating the paper was retracted. However we did not remove these entries or determine how many of them existed in the data. For the Lucene Index, we utilized the StandardAnalyzer which performs tokenization, lowercasing of terms, stop word removal, Porter stemming, and normalization of fields [4]. We indexed all fields in the data set as TextFields but added additional index fields to store the original values of the Title and Abstract fields to make document recovery easier for later analysis.

For comparison, we utilized a vectorization model together with Annoy [1] to index our data set. We originally intended to use the Gensim doc2vec model [5] for vector generation but could not get this model to generate remotely relevant results for our queries. We are unsure if this was due to the data set or if we did not have adequately chosen hyperparameters for the doc2vec model. We did not have sufficient time or compute resources to investigate further. As an alternative we went with the **all-MiniLM-L6-v2** Sentence Transformer [3] implemented in the HuggingFace Machine Learning library. Due to the length constraints common to transformer models on the input embeddings, we only generated embeddings for the TItle and Abstract fields and used the Gensim **remove_stopwords** helper to further reduce the length prior to encoding. Despite these measures we still had to enable truncation when encoding the content. We attempted to use the Longformer transformer [7] in HuggingFace to avoid having to truncate the encoding, but had difficulty finding documentation on how to interpret the encoding output to use the embeddings with Annoy. After embedding generation, we extracted the CLS embeddings and applied a mean pooling operation [6] to flatten the embedding and then normalized the vectors for use with Annoy. The embedding vectors were then mapped to an

Annoy index with 200 trees. Due to compute constraints, we were unable to try an Annoy index with more than 200 trees and smaller tree sizes like 50 were not producing relevant results.

The full code can be found in the GitHub repo at https://github.com/elliot-j/csc-583-search-project. The repo's README.md file contains full instructions on how to build and run the code. The Python code has specific hardware requirements which are further detailed in README and may not run correctly on a machine with insufficient RAM.

Query and Scoring Analysis

For querying the Lucene index, we used the MultiFieldQueryParser to enable simultaneous querying of the Title and Abstract fields and scored using BM25 and tf-idf similarities. Queries for the Annoy model were scored using the Angular similarity setting in the model which computes a cosine distance, which was then converted to a cosine similarity score as described in [8]. Because the Arxiv data set has not been extensively used for information retrieval, we created 31 queries (see Appendix 1) from topics covering computer science, astronomy, physics, and medicine. Arxive does not normally host papers related to medicine and biosciences, which are typically hosted at the sister site bioRxiv, so the medicine queries were added to give contrast to the results. Inspiration for some queries was drawn from articles from the science journalism organization Quanta Magazine. Because Annoy does allow easy retrieval of the returned results, we took the docID's returned by Annoy and fed them back into Lucene to trace back the original document.

In order to evaluate the quality of the results returned by the two IIR systems (Lucene and vector-based indexing and search), we calculated the MRR scores and compared the two scores against one another. MRR measures the position of the first relevant item in the ranked list of results returned by the IIR system and then takes the reciprocal of that position. The reciprocal rank is then averaged across all queries. We performed this evaluation for all 31 queries for both IIR systems.

We used the precision at MRR (Mean reciprocal rank). We included the precision of the entire query because in the context of the question it's the proportion of answers that were correct over the entire collection of questions that are generated by us. We wanted to check the entire answer if it predicted the actual answer based on the entire query.

Measuring Performance

We used precision at **MRR (Mean reciprocal Rank)** and **NDCG (Normalized Discounted Cumulative Gain)** to assess our project's performance. MRR enables us to gauge how well the choses query method is at ranking a document where a user is most likely to select it, while NDCG enables us to examine which method is better at surfacing related content that a

user might be interested in looking at while also encapsulating how well the model ranks the supporting relevant results

Since we did not analyze this dataset extensively beforehand, we reviewed the results returned by Lucene and Annoy to identify a document in the results that best fit each query. At the same time, we also gave each document in the results a relevance score of [0,3] with 0 being not relevant and 3 being highly relevant. For the ensemble model, through experimentation we found that a lambda of 36.1 with the BM25 Scoring and Annoy produced the best results. The data used to generate the below figures can be found in Appendix 2

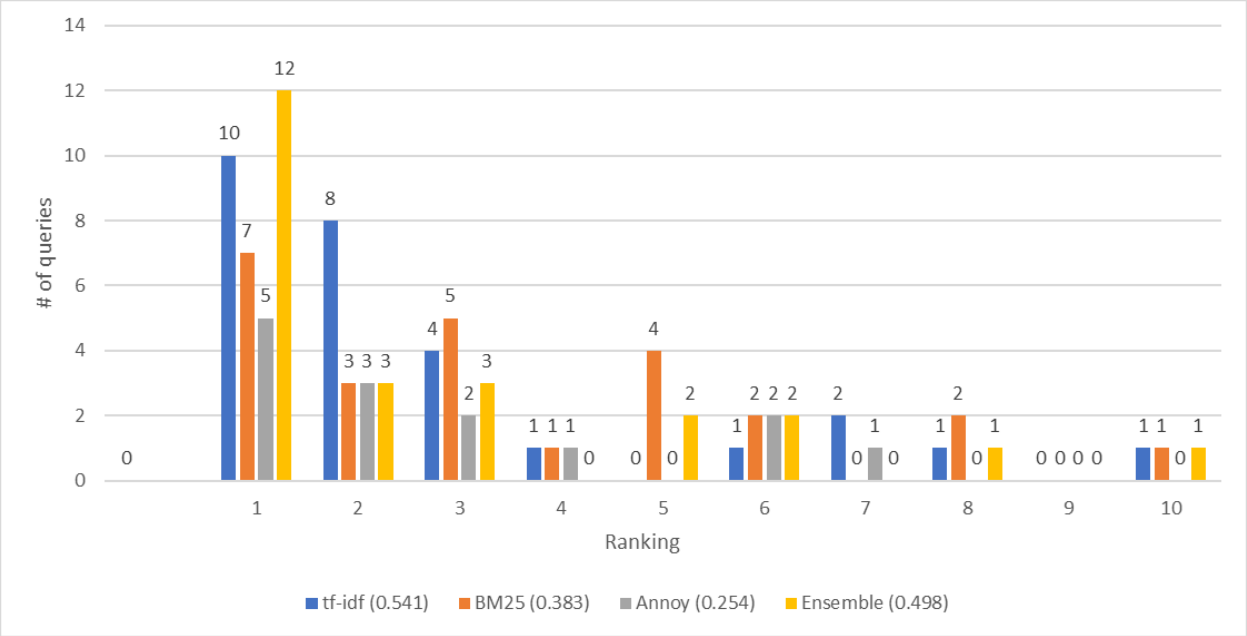

Figure 1) Ranking of best fit document by search type

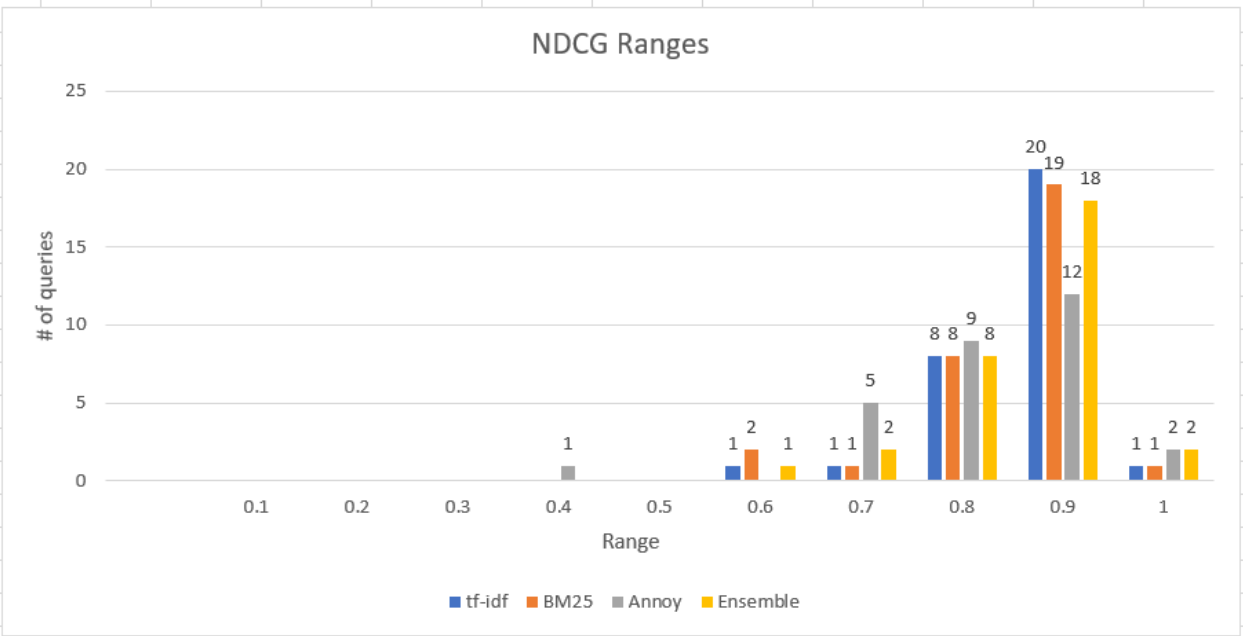

Figure 2) Range Counts of NDCG Scores

	Tf-idf Ranking	BM25 Ranking	Annoy Ranking	Ensemble (BM25 + Annoy)
MRR Average	**0.541**	**0.383**	**0.254**	**0.498**
NDCG Average	**0.900**	**0.897**	**0.814**	**0.906**

Table 1) Average Scores of metrics by Search Type

From Figure 1 and Table 1, we can see that the ensemble method is better at ranking the best fit document first for a given query (12 vs 10). We also see that the Average MRR for TF-IDF is slightly higher than ensemble, though this difference (.043) is small enough that we consider the two method equivelent. Additionally TF-IDF scoring is better at placing the document of interest within the top 3 results than ensemble (22 vs 18). Interestingly, Annoy does very poorly on its own ranking the document of interest in a favorable position.

In Figure 2, the Annoy model still comes in last place for returning relevant content. It should be noted that only documents in the Annoy results were also returned by either Lucene scoring method. With the average NDCG score for annoy sitting at 0.814, it is clearly returning relevant results that would have otherwise been missed by Lucene. The ensemble model has the best performance overall, but the difference between Ensemble and TF-IDF and BM25 is so

small that we consider the three equivalent. As we detail in the Error Analysis section, there are many opportunities to improve Annoy performance that could easily boost the Ensemble model performance on both metrics.

Errors analysis

After experimenting with two search strategies and evaluating MRR, the following inferences can be drawn:

1. **MRR for Lucene using BM25 and Lucene using tf-idf** - we observed that BM25 did not perform well across all queries. On the other hand, tf-idf yielded good results, with an MRR performance score of 0.541 compared to BM25's score of 0.383. Since we left the BM25 hyperparameters at their defaults, this could likely be improved greatly with better values. Another possible issue is that because we examined the tf-idf results first, this biased our selection of the document of interest giving an unfair advantage to TF-IDF.

2. **Lucene using tf-idf similarity and Annoy** - we observed that Lucene using tf-idf similarity outperformed Annoy with an MRR performance score of 0.541 compared to Annoy's score of 0.254. Due to compute resources we were limited to having only 200 trees in the Annoy graph. Since our data set has 2.2 million records, this tree count was likely the biggest contributor in Annoys poor MRR results. Another possible contributor to the poor annoy performance was the embedding size limitations, since we were truncating embedding vectors to meet size constraints of the sentence transformer. A transformer with a larger embedding limit may result in a more positive outcome than the one we focused on. Lastly, it is also possible that the queries we came up with for this experiment were more suited to Lucene searches than Transformer+Annoy searches

3. **Lucene using tf-idf similarity, BM25, Annoy and ensemble method** - we observed that Lucene using tf-idf similarity outperformed all of the other methods as MRR performance score of annoy is 0.254 , BM25 of 0.383 and ensemble method of 0.498 which is clearly less than the MRR performance score of 0.541.

For our dataset, we noted that we can’t just rely on Annoy for indexing and searching. It must be used in combination with traditional non-neural based indexing and searching, like Lucene. This combination yielded better results, and we observed that Lucene (with tf-idf) was able to answer certain straightforward queries that Annoy was not able to detect. By leveraging the strengths of both techniques, we can achieve a more robust and accurate search system.

Classifying issues:

1. The first issue we have identified which is important because when we are going through the performance analysis we have faced it a lot and come out to the point that It has been observed that the dataset is significantly skewed towards tf-idf. The tf-idf weighting scheme is being disproportionately represented in the dataset as compared to other weighting schemes. This skewness potentially result in suboptimal performance of the system, as it does not account for the unique characteristics of other weighting schemes that are present in the dataset.
   As a result, we have concluded that it is imperative to conduct a thorough evaluation of the dataset to ensure that it is representative of all the weighting schemes and that the system's performance is not adversely affected.

2. The second issue faced is which involves the suboptimal performance of the standalone Annoy system, as it initially yielded unsatisfactory results. However, upon implementing a stemming process, the system exhibited a marked improvement, generating more relevant and pertinent outcomes. This enhancement in performance highlights the significance of incorporating appropriate preprocessing techniques to ensure the desired system output.

3. Lastly the issue and also the limitation we have faced at hand pertains to the insufficient training of our system, which can be attributed to the limitations in available resources. Constructing Annoy indexes demands a significant amount of resources and time, which unfortunately, all team members are not equipped to handle in terms of their individual resource capabilities. As a consequence, the system's performance has been adversely affected.

Lesson Learned

Although the problem addressed by this project did not address a cutting edge problem in information retrieval, it proved to be an invaluable experience in the process of building and evaluating the performance of a search engine using a domain specific data set. Many businesses need to search through internal documentation or data as part of their ongoing operations but may not have the resources or budget to integrate with cloud hosted search services like AWS Elasticsearch. The data set used in this project is not a standard data set for IR research and was not extensively analyzed before hand, simulation a real world scenario where a development team would be asked to build a search capability onto a data set they have no prior knowledge of while being able to determine if the solution they have build is capable of meeting business needs.

References

1. **Spotify Annoy for Approximate Nearest Neighbors** https://github.com/spotify/annoy
2. **Cornell University Arxiv.org Abstract Collection** https://www.kaggle.com/datasets/Cornell-University/arxiv
3. **MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers (2020); Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, Ming Zhou;** https://arxiv.org/abs/2002.10957
4. **Preprocessing provided by Lucene StandardAnalyzer** https://stackoverflow.com/questions/17011854/whats-the-difference-between-lucene-standardanalyzer-and-englishanalyzer
5. **Introduction to Gensim Doc2Vec** https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html
6. **Mean Pooling on Sentence Transformer** https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
7. **Longformer: The Long-Document Transformer (2020); Iz Beltagy, Matthew E. Peters, Arman Cohan;** https://arxiv.org/abs/2004.05150
8. **Convert Annoy Cosine Distance to Cosine Similarity** https://github.com/spotify/annoy/issues/530
9. **Install Annoy in Python Windows environment** https://www.programmersought.com/article/95834605670/

Appendix 1 - Queries for analysis

1. dark matter and black holes AdSn (anti-desitter space)
2. room temperature super conductor
3. quantum computer supremacy
4. spooky action at distance and quantum entanglement and encryption
5. shortest path on negative graph weights
6. neural network hallucination prevention and mitigation
7. performance analysis of quantum annealing
8. gate all around transistors vs FinFET
9. number theory modular form functions noncongruence and congruence
10. intransitive dice rolling
11. natural language processing and deep learning
12. types of the data mining
13. optimization algorithms
14. GDP growth and inflation rate
15. power electronics applications
16. what is fiscal policy
17. speech recognition and audio signal processing
18. genome sequencing of covid-19
19. different sampling methods
20. laws of thermodynamics
21. great barrier reef
22. what is bias in machine learning
23. Ocean acidification
24. Shark antibodies antiviral therapies
25. what is CPI and GDP
26. genomics of adaptive evolution
27. Molecular hydrogen clouds
28. Types of cancer
29. Human beings evolution
30. expansion of the falling factorial
31. What is the limitation of quantum mechanics

Appendix 2 - Query Result Evaluation Data

MRR Scores

Best Fit Document	Tf-idf Ranking	BM25 Ranking	Annoy Ranking	Ensemble (BM25 + Annoy) Lambda (36.1)
1 [133869]	2	5	0	6
2 [1945581]	1	5	2	1
3 [1027458]	1	0	0	1
4 [1218574]	10	10	0	10
5 [205807]	1	1	1	1
6 [1046961]	1	1	0	1
7 [1410596]	2	3	0	3
8 [1346123]	7	2	1	0
9 [2154967]	1	1	0	1
10 [480172]	1	5	0	5
11 [1251555]	2	3	2	0
12 [517193]	3	6	0	6
13 [1658676]	10	6	3	0
14 [361214]	7	4	7	1
15 [1636233]	1	1	1	1
16 [468086]	2	0	0	0
17 [1118514]	1	1	0	1
18 [1255935]	3	0	1	1

19 [1373649]	8	5	0	5
20 [267908]	1	1	6	1
21 [1014444]	2	3	0	3
22 [1171904]	2	3	3	2
23 [1661826]	2	2	2	1
24 [1697177]	6	0	0	0
25 [93477]	3	8	0	8
26 [62262]	0	8	6	0
27 [1824309]	2	0	4	0
28 [1174207]	3	3	0	3
29 [1584628]	4	2	0	2
30 [521466]	1	1	0	2
31 [2205489]	0	0	1	1
Average	0.541	0.383	0.254	0.498

Normalized DCG

Query Number	Annoy	BM 25	TF-IDF	Ensemble (BM25 + Annoy)
1	0.779	0.914	0.937	0.757
2	0.867	0.761	1	0.921
3	0.884	0.979	0.946	0.991
4	0.938	0.926	0.92	0.958
5	0.95	0.939	0.934	0.938
6	0.832	0.88	0.982	1.0

7	**0.914**	**0.894**	**0.973**	**0.898**
8	**0.97**	**0.959**	**0.958**	**0.857**
9	**0.771**	**0.866**	**0.818**	**0.873**
10	**0**	**0.978**	**0.969**	**0.977**
11	**0.948**	**0.912**	**0.945**	**0.908**
12	**1**	**0.815**	**0.828**	**0.807**
13	**0.741**	**0.915**	**0.896**	**0.912**
14	**0.966**	**0.926**	**0.952**	**0.965**
15	**0.947**	**0.983**	**0.959**	**0.994**
16	**0.836**	**0.88**	**0.847**	**0.845**
17	**0.948**	**0.901**	**0.926**	**0.905**
18	**0.97**	**0.912**	**0.812**	**1.0**
19	**0.959**	**0.909**	**0.979**	**0.961**
20	**0.734**	**0.944**	**0.952**	**0.952**
21	**0.919**	**0.937**	**0.921**	**0.935**
22	**0.862**	**0.85**	**0.817**	**0.855**
23	**0.878**	**0.986**	**0.966**	**0.938**
24	**0**	**0.984**	**0.729**	**0.984**
25	**0.817**	**0.926**	**0.991**	**0.925**
26	**0.827**	**0.895**	**0.945**	**0.891**
27	**0.908**	**0.885**	**0.922**	**0.881**
28	**1**	**0.932**	**0.895**	**0.93**
29	**0.717**	**0.635**	**0.673**	**0.635**

30	**0.484**	**1**	**0.956**	**0.776**
31	**0.886**	**0.606**	**0.818**	**0.906**
Average	**0.814**	**0.897**	**0.90**	**0.906**