Neural Inverted Index for Fast and Effective Information Retrieval using Differential Search Index

Syed Saad Hasan
Sapienza University of Rome
hasan.2106512@studenti.uniroma1.it

Abstract

In the field of information retrieval (IR) the crucial factor is in the capability of systems to provide relevant rankings of documents in response to user queries. Typically, traditional information retrieval (IR) systems use a sequential process that includes indexing and then retrieval. Our research introduces a new approach that is in line with the previously proposed Differentiable Search Index (DSI) framework. This approach integrates the processes of indexing and retrieval into a cohesive Transformer language model. We performed studies using several methods to train the DSI utilizing an alternative configuration for the auto-regressive mode. The main differences between our model and other similar models are the decreased probability of using teacher forcing which better simulates the process of generating document IDs (docids) and the limitation of generation to existing docids during inference through the utilization of a specialized data structure. The aim of our research is to develop a unified model denoted as 'f' which operates as a sequence-to-sequence architecture. This model will have the ability to evaluate a query 'q' and generate a suitable docid for that query in an auto-regressive manner. When drawing conclusions, we generate a list of document IDs that are arranged in order of significance. This is accomplished by implementing a limited beam-search algorithm on the model's outputs. This approach is designed to mimic the functionality of an index created from a set of documents and is used to retrieve relevant content. The training and evaluation of our model depend on the MS MARCO dataset which has been augmented using the Pyserini package. To assess the efficiency of our model we explicitly analyze three performance metrics: Mean Average Precision (MAP) Precision@10 and Recall@1000.

1 Introduction

Given the challenging nature of the issue and the limited computer resources at our disposal we explored several approaches to achieve significant outcomes in a more advanced training environment. We performed tests using several methodologies in three essential areas: pre-processing data representation and modeling (including architecture and inference). This allowed us to identify the most effective strategies for improving performance.

2 Dataset

To optimize the process of generating datasets for training and testing different models we first establish two dictionaries: one for queries and another for documents. The raw text of all MS MARCO queries and documents is stored in dictionaries with their IDs serving as keys. Every item in the query dictionary also contains a roster of pertinent documents. During the construction of these dictionaries, we simultaneously generate a corpus that encompasses all the processed raw data. The objective of this corpus is to use a Word2Vec model for generating embeddings of a certain length (EMBEDDING SIZE) by using a window size of MAX TOKENS. The Word2Vec model after being trained produces two kinds of embeddings for each query and document:" e m b " (which is the average of all word embeddings) and" first L emb" (which is the concatenation of the first MAX TOKENS word embeddings). By using these two dictionaries we generate two datasets that are specifically customized for training Siamese networks. The QueryDocumentDataset comprises triplets that include a query a document and a relevance value. The relevance value is assigned a binary value of 1 if the document is considered relevant to the inquiry and 0 if it is not. The TripletQueryDocumentDataset generates triplets consisting of an anchor a positive instance and a negative instance. The anchor serves as the query the positive example is a relevant document and the negative example is a randomly selected unrelated website. In addition, we generate two supplementary datasets to train the Seq2Seq model using the same dictionaries for queries and documents. The DocumentDataset provides encoded documents and their accompanying encoded document IDs whereas the RetrievalDataset provides encoded queries and their respective encoded document IDs. We generate document and query encodings via the T5-small tokenizer. When choosing tokens for documents we choose for the first 32 tokens. However, for inquiries we just choose the first 9 tokens. When there is a scarcity of tokens we handle them appropriately. To enhance the efficiency of document identification vocabulary (docids) we use individual numerical digits (0-9) as tokens. The tokens are comprised of three distinct values: 12 represents the beginning 10 represents the conclusion and 11 is used to complete any missing elements in the sequence.

3 Baseline Performance Analysis of Siamese Networks

To establish baselines for comparison in this project we first modeled the task using a different approach. We aimed to learn the relevance between documents and queries by directly comparing their contents without using docids until the final retrieval phase. The models discussed in the following section are designed to output a relevance score between an input document and query. To achieve this, we compute the relevance score in various ways by using the feature vectors of the query and document passing them through the same network. Consequently, at inference time we need to iterate over all the documents passing each one along with the query through the model. This leads to a higher retrieval time even though the training time is shorter compared to the DSI model.

3.1 SiameseNetwork Baseline

The first tested baseline SiameseNetwork processes the embeddings (emb) of queries and documents out- putting the probability of their correlation by concatenating the two feature vectors and passing the final vector into a fully connected layer followed by a sigmoid function. Despite achieving high accuracy levels (up to 87%) the model's performance on the Precision@k metric reveals its inability to effectively distin- guish between relevant and random documents often assigning high similarity scores to irrelevant documents. This is evidenced by the Precision@10 and Recall@1000 scores reported in Table 1 which show that relevant documents do not consistently have the highest scores and instead appear in lower-ranking positions.

3.2 SiameseTransformer Baseline

The second baseline SiameseTransformer utilizes the first MAX TOKENS embeddings (first L emb) of queries and documents. This model directly uses the embeddings obtained from the tokens of the documents and queries to compute attention scores for building the two feature vectors. Like SiameseNetwork it generates a relevance score using Cosine Similarity. However, this model also struggles to differentiate between relevant and random documents frequently attributing high scores to both.

3.3 SiameseTriplet Baseline

The final baseline SiameseTriplet was trained using TripletMarginLoss providing positive and negative ex- amples corresponding to the queries. This method leverages the principles of contrastive learning and showed notable effectiveness over previous baselines. Unlike earlier methods the contrastive approach in SiameseTriplet effectively discerns between similar and dissimilar query-document pairs. It optimizes the embeddings so that similar pairs are pulled closer while dissimilar ones are pushed apart creating a distinct margin imposed by the triplet loss function. This significantly improved the model's ability to generalize from training data to unseen queries as shown in Table 1. Figure 1 depicts the ranked documents for a random test query.

Doc ID	Score	Relevant
5763747	0.9248231649398804	0.0
775666	0.9046399593353271	0.0
6702148	0.8990009427070618	0.0
5347376	0.8880547881126404	0.0
4019802	0.8878763318061829	0.0
354963	0.8771447539329529	1.0
1672345	0.8759815692901611	1.0
8291243	0.8745550513267517	0.0
7367184	0.8597170114517212	0.0
767321	0.8580762147903442	1.0

Figure 1: In the top 10 retrieved documents, 3 of them are relevant ones.

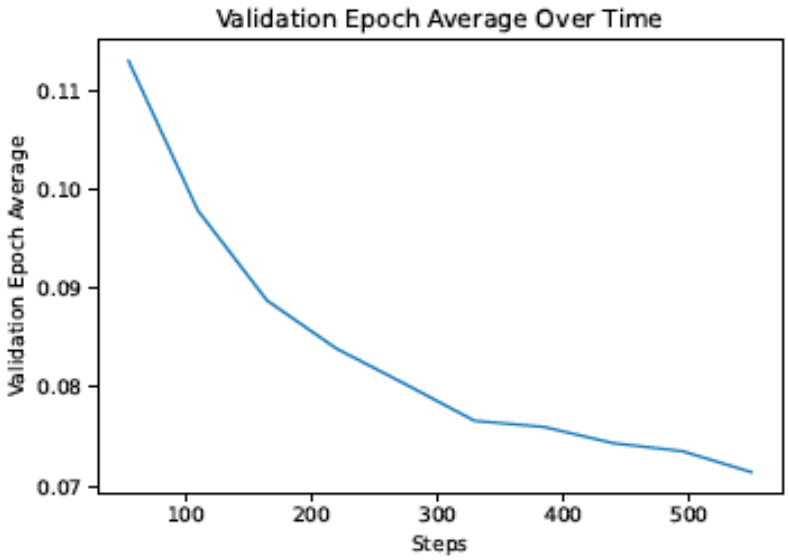

Figure 2: Validation loss during contrastive learning

4 DSI Transformer-Based Model

To reduce the risk of overfitting the documents we chose a lightweight sequence-to-sequence model for the task. We built two different embedding layers: one for the input sequence and one for the target sequence as they are obtained by two different tokenizers. Our model has three encoder-decoder transformer layers and a final fully connected layer. The model outputs a probability vector of size docid vocab size for each element in the sequence and each element in the batch. In summary the output shape is (seq len, batch size, docid vocab size).

4.1 Training Setup: Teacher Forcing and Auto-Regressive

Our training setup has two main sections: indexing and retrieval. During indexing the model learns the documents' knowledge by training on the entire corpus to map every document to its docid without splitting the dataset. During retrieval the model generates the entire target sequence from a special start token given a query. For retrieval learning we fine-tuned on (query, docid) pairs and split the dataset into training, validation, and test sets. Initially we used simple teacher forcing feeding the decoder with the target sequence up to the last token and computing the loss with the target sequence shifted by one. However, this approach failed because the model couldn't generate target tokens without ground truth. We then modified the training step to use teacher forcing with a decreasing probability (starting at 1) that decreases every epoch. When not using teacher forcing the model enters an auto-regressive setup: it generates one token at a time by choosing the arg max of the current token's prediction feeding the decoder with the last generated token at each iteration. However, this approach also led to performance drops because errors propagated through the sequence. Final solution was to modify the auto-regressive mode to feed the decoder with all generated sequences up to the current token. We used this mode only for retrieval decreasing the probability of teacher forcing at each validation epoch by 0.003.

4.2 Hyperparameter Optimization

After many attempts we found the best combination for this training to be embedding size of 120 three transformer layers a dropout rate of 0.2 and a learning rate of 0.0005.

4.3 Results

Due to Google Colab's limitations (RAM saturation and GPU usage) we couldn't train for more than 400 epochs for indexing and another 400 epochs for retrieval. With more powerful computational resources we could achieve higher results. The plots in Figures 2 and 3 show promising results with our setup. We trained until RAM was saturated and then

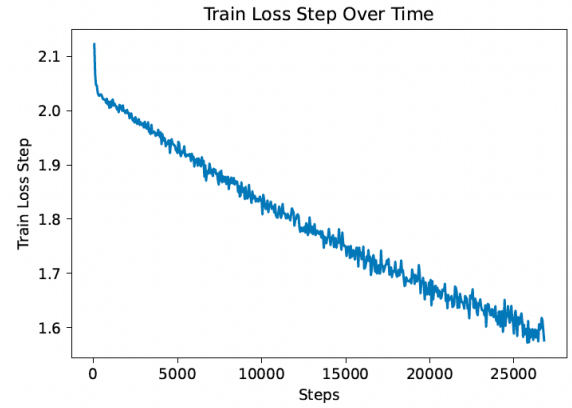

Figure 3: Train Loss during the memorizing train

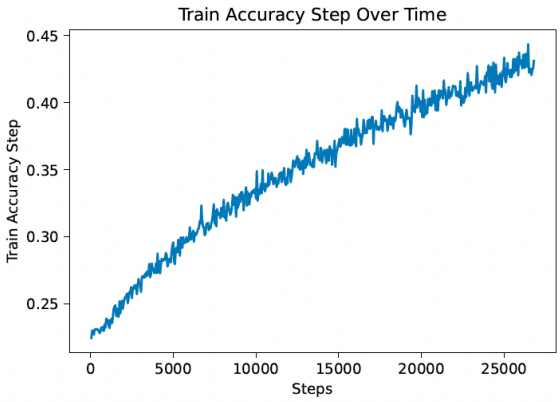

Figure 4: Train Loss during the memorizing train

During the fine-tuning phase, we observed initial high variability in the loss, as shown in Figure 5. This variability stabilized as the training progressed, indicating effective learning and adaptation to the retrieval task. The accuracy, depicted in Figure 6, shows a positive trend in the model's ability to correctly associate queries with document identifiers. The validation loss, shown in Figure 7, initially decreases but then rises slightly, reflecting the gradual reduction of teacher forcing. In contrast to the validation loss, the accuracy on the validation set, depicted in Figure 8, shows a consistent increase. This implies that despite the increase in validation loss, the model's predictions are becoming more accurate over time, suggesting that the model is increasingly better at identifying the correct document identifiers.

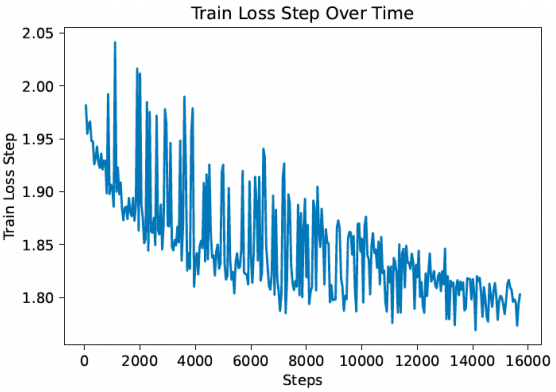

Figure 5: Train loss during retrieval training. It can be seen that the noise due to the reduction of teacher forcing decreases over time

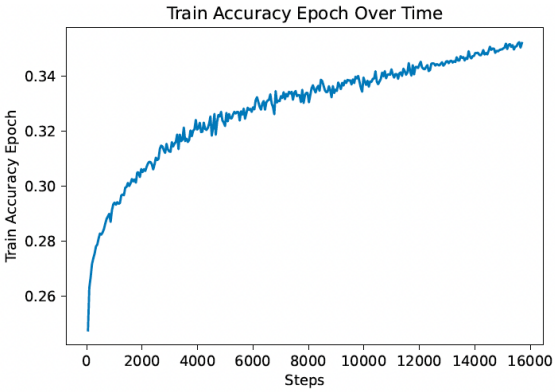

Figure 6: Train accuracy during retrieval training

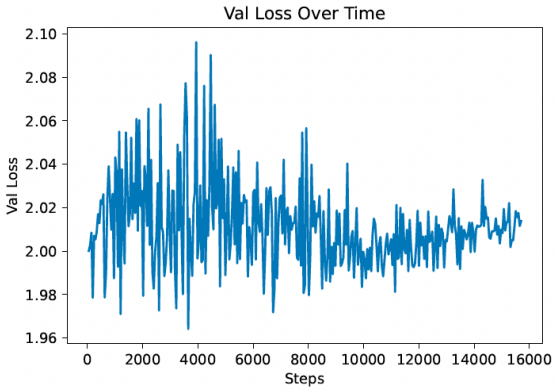

Figure 7: Validation loss during retrieval training starts to decrease and then rise up, because of the reduced capacity of the model and the reduction of teacher forcing

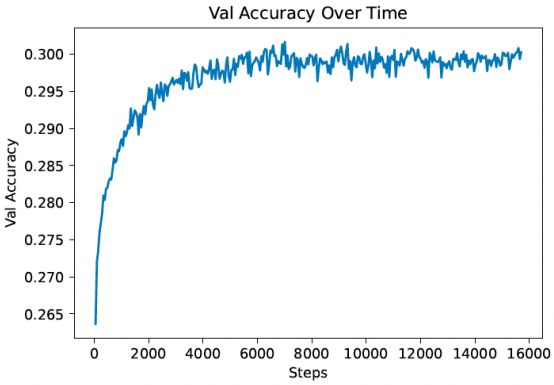

Figure 8: The validation accuracy surprisingly increases during the retrieval training.

5 Metrics and Inference

During the inference method we imposed a constraint on the model stipulating that it could only produce document identifiers (docids) that already existed. To accomplish this task a particular data structure known as a ”Trie” was used. One of the features included in the Trie data structure is the get next tokens function. This function accepts a collection of tokens as input and then provides the potential subsequent tokens that may be used in the generation of a valid docid. The top k beam search function has recently been enhanced to provide a search strategy that is limited to beams. By using this technique sequences are produced that adhere to the limitations specified by the Trie. The technique starts by assembling hypotheses where each hypothesis comprises a start-of-sequence token and an initial log probability of 0. Subsequently the procedure proceeds to the subsequent stage. At the start of each cycle the algorithm modifies each hypothesis using a maximum sequence length. If an end-of-sequence token is used to terminate a hypothesis the hypothesis will be preserved. Alternatively, the model may calculate the likelihood of future potential tokens with the Trie data acting as the constraining element. By considering this we guarantee that on Figure 4: Train loss during retrieval training. The noise due to the reduction of teacher forcing decreases over time authentic expansions of the existing sequence are considered. Subsequently additional tokens are appended to each hypothesis resulting in a modification of the probabilities linked to those hypotheses. The program evaluates all potential extensions and then chooses the top k hypotheses based on their likelihood of being accurate.

Table 1: Models Training and Validation/Test Scores

Model	**MAP**		**Precision@10**		**Recall@1000**	
	Train	Val	Train	Val	Train	Val
SiameseNetwork	0.060	0.000	0.020	0.000	0.240	0.420
SiameseTransformer	0.000	0.000	0.000	0.000	0.039	0.100
SiameseContrastive	0.302	0.213	0.158	0.158	0.783	0.778
Seq2Seq	0.192	0.190	0.172	0.168	/	/

6 Conclusion

The objective of this project was to combine the indexing and retrieval processes into a unified Transformer language model departing from the conventional approach of using separate architectures in information retrieval (IR) systems. The rationale for our method was the notion of the Differentiable Search Index aiming to amalgamate these distinct techniques into a more efficient adaptable and seamless retrieval process. By conducting extensive testing on the MS MARCO dataset and using various training tactics such as auto- regressive and teacher-forcing procedures we have shown the practicality and promise of this integrated model. The findings of our experiment demonstrated enhanced retrieval accuracy in comparison to traditional self- supervised models as shown by metrics such as Mean Average Precision and Precision@10. In addition, we saw that progressively reducing the dependence on instructor forcing throughout the training process the network to produce sequences with higher levels of accuracy. This method surpassed the performance of Siamese networks and even outperformed standard sequence-to-sequence models that only anticipate one missing character. Nevertheless, there are still obstacles that need careful consideration. The limitations in processing resources constrained the scope of our training indicating that using more powerful setups may lead to more substantial advancements. By systematically reducing the need for teacher assistance throughout training we gained valuable insights into the learning process of the model. This in turn revealed potential areas for future improvement. To summarize this comprehensive method has significant promise for many information retrieval fields facilitating the development of systems that are more effective precise and user oriented.

7 References

- Tay, Y., Tran, V. Q., Dehghani, M., Ni, J., Bahri, D., Mehta, H., Qin, Z., Hui, K., Zhao, Z., Gupta, J., Schuster, T., Cohen, W. W., Metzler, D. (2022). Transformer memory as a differentiable search index.
- Bajaj, P., Campos, D., Craswell, N, Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitr B., Nguyen, T., Rosenberg, M., Song, X., Stoica, A., Tiwary, S., Wang, T. (2018). Ms marco: A human generated machine reading comprehension dataset.
- Lin, J., Ma, Y., Lin, S., Yang, H., Lin, C. (2021). Pyserini: An Easy-to-use Python Toolkit to Support Replicable IR Research with Sparse and Dense Representations.
- Mitra, B., Craswell, N., Mitra, B. (2018). Deep Learning for Information Retrieval: Fundamentals and Advance