# TRANSFORMER MEMORY AS A DIFFERENTIABLE SEARCH INDEX USING SCHEDULED SAMPLING

**Valerio Di Stefano**
Sapienza University of Rome
distefano.1898728@studenti.uniroma1.it

**ABSTRACT**

In 2022, a paper by Google Research titled "Transformer Memory as a Differentiable Search index" demonstrated that information retrieval can be accomplished with a single Transformer, in which all information about the corpus is encoded in the parameters of the model, which acts as a Differentiable Search Index (DSI). The proposed text-to-text model maps string queries directly to relevant document IDs, answering queries directly by using only its parameters, which simplifies the retrieval process. We present results of a novel approach for the training of a DSI transformer architecture which employs a scheduled sampling technique to mitigate exposure bias and does not rely on clustered or semantic document IDs.

## 1 Introduction

This paper explores a novel information retrieval (IR) framework that utilizes a differentiable function to generate a sorted list of document identifiers in response to a given query. The approach is called Differentiable Search Index (DSI), and was originally proposed in the paper "Transformer Memory as a Differentiable Search Index" by researchers at Google Research [1]. DSI aims at both encompassing all document's corpus information and at executing retrieval within a single Transformer language model, instead of adopting the traditional "index-then-retrieve" pipeline used in most modern IR systems. The paper presents the implemented solution, a Sequence to Sequence transformer model that, given a natural language query as input, directly returns a list of document IDs ranked by relevance to the query. The proposed solution combines the DSI approach with the Scheduled Sampling technique for Transformers, inspired by the similar technique described in the paper "Scheduled Sampling for Transformers" by Mihaylova et al. [2]. We evaluate the performance of the proposed models using the "Mean Average Precision" (MAP) and the "Recall at K" metrics computed on the MS MARCO dataset [3], and we compare the results with several baselines (TF-IDF, Word2Vec, Siamese Network and also other traditional Transformer approaches).

## 2 Data

The MS Marco dataset used to train and test the various models was simplified to only contain short passages of documents, and only include 1000 total queries with a list of 10 relevant documents each. The final set of documents used throughout the experiments was comprised of only documents relevant to at least one query. Ultimately, the final number of documents used was less than 10.000, since some of the considered queries have overlapping relevant documents, while the number of relevant documents for some of the queries was higher than 10, because of some of the included documents being relevant to both said queries and also one or more other queries. Documents and queries were stored in two separate dictionaries, and include their IDs, the raw text content, and the corresponding vector embedding computed using a Word2Vec model trained on the corpus of documents (see Section 3). The queries also contain the list of relevant document IDs in the final dataset. An ID from 1 to N, with N equal to the total number of documents in the dataset, was also assigned to each document, while for queries, the original ID in the MS Marco dataset was retained. Before assigning their corresponding new IDs, documents got shuffled, so that no semantic is assigned to each document ID, to test out a DSI IR framework in which, when new documents need to be added to the database, there is no need to re-compute (and therefore re-assign and maintain) new IDs for all documents to maintain the semantic without breaking the uniqueness property of each identifier.

### 2.1 Siamese Network Dataset

A separate dataset was built for training the Siamese Network model used as a baseline (see Section 3). This dataset was composed of <query, positive document, negative document> triplets, where "query" is the vector embedding of a query's text, while "positive document" and "negative document" are vector embeddings of, respectively, the text of a document relevant to that query (for each of the relevant documents of the query itself) and the text of a document that is not in the query's relative documents list. These vector embeddings are the ones stored in the documents and queries dictionaries, in turn computed using the Word2Vec model described in section 3. This dataset was used to train the Siamese Network model using a "Triplet Loss" following a contrastive learning approach.

### 2.2 Transformer Datasets

Two separate datasets were also built for training the "DSI" Transformer models. The first dataset was built for the "indexing task" (see section 4), and was composed of <document text, document ID> tuples: both the items of the tuple are tokenized, respectively using a pretrained english-words tokenizer (with a padded maximum token length of 64 tokens), and an ad-hoc tokenization approach which considers digits from 0 to 9 for each document ID and also adds to the tokenized document ID sequence three special tokens for the start of sequence, the end of sequence, and padding (to the maximum document ID length). The second dataset built for the Transformer training was used in the "retrieval task" (see section 4), and is composed of <query text, document ID> tuples, once again with tokenized items (the same approach described above was used).

## 3 Baseline Models

To evaluate the final transformer models with respect to appositely defined baselines, three different models were also implemented: a TF-IDF vectorizer, a Word2Vec model, and a Siamese Network model.

### 3.1 TF-IDF

The first baseline consists of a simple TF-IDF vectorizer (Term Frequency & Inverse Document Frequency), which uses no Machine Learning for its training, but rather simply computes the TF-IDF scores for each word in the document corpus and each document, necessary to then compute the "Mean Average Precision" and "Recall at K" metrics. This model is evaluated by computing the TF-IDF score for each query given as input and each term in the document corpus, then selecting the top K documents ranked by this score.

### 3.2 Word2Vec Model

The second baseline built to evaluate the final Transformer model consists of a Word2Vec model which maps (tokenized) text into a vector of size 128. This model was trained on the entire corpus of documents in the dataset, and was also used to compute vector embeddings to store in the documents and queries dictionaries. At inference time, this model uses the traditional "indexing then retrieve" pipeline, and thus, for each (tokenized) text query given as inputs, computes the corresponding vector embedding and then cycles through each document in the database to compare the corresponding vector embeddings using cosine similarity and later on select the top K documents based on this value.

### 3.3 Siamese Network Model

The third baseline model consists of a Siamese Network model trained using a Triplet Loss approach, and in particular using the corresponding model's dataset of triplets described in section 2. This model takes as input two vector embeddings (of a certain text) and outputs a similarity score between the two vectors. At inference time, in order to compute the K most relevant documents as a result of a text query given as input, the vector embeddings of all the documents in the database are used as the second input to the model, and thus the computed similarity scores are used to rank the documents and thus retrieve the K best results. Once again, this approach can be considered an "indexing then retrieve" approach, different and also slower than the DSI approach using Transformer models described in the next section.

## 4 Transformer Models

The implemented DSI Transformer model, which uses 4 multi-head attention heads and 3 hidden layers for both the encoder and decoder modules (having input embeddings of size 128) was trained using three different approaches,

later evaluated and compared using the Mean Average Precision and Recall at K metrics (see section 5): a "teacher forcing" approach, a pure "auto-regressive" approach, and an approach based on "scheduled sampling" for Transformers (inspired by Mihaylova et al. [2]).

All three training approaches were split into two phases: a training for the "indexing" task, in which, starting from the (tokenized) text content of each document, the model learns how to directly map document IDs to documents, and a second training phase for the "retrieval" task, in which, instead of documents, the model is given <query, relevant document ID> pairs (both tokenized, as discussed in section 2) and thus learns to directly output document IDs in response to queries, using the knowledge acquired during the "indexing" task.

In the training for the "indexing" task, no split of the training data was used, but instead the entirety of the training data, hence all of the <document text, document ID> pairs (described in section 2), was used for the training set: this ensured that the model could learn to map document texts to document IDs for all of the documents in our corpus instead of learning to associate text to appropriately constructed IDs (which is not what we want, since no semantic is associated to document IDs in our approach).

At inference time, the Transformer models trained with the different approaches all use an "auto-regressive" approach to generate the output tokens (hence the document ID digits): the model starts from the (tokenized) query text as input sequence to the Transformer's encoder module, and the "start of sequence" document ID token, with the remaining target tokens being masked, as input to the Transformer's encoder; the model then takes the top N tokens at each new auto-regressive step (each new output token generation), with N depending on the number of final sequences to generate in the output ranking.

An important note for the training of the Transformer models is that, unlike the usual machine learning or deep learning models' training settings, during the "indexing" task, we try to "overfit" our model to the training data: in this setting, in fact, we want to memorize the training data (the documents corpus) in the weights of the Transformer model's network, to then be able to perform inference (generate document IDs from documents or query texts) without accessing the document corpus again.

### 4.1 Teacher Forcing Transformer

The first training approach tested for the DSI Transformer model was based on "teacher forcing": during training, the model is given, as input, both the source sequence (a tokenized query text) and the tokens of the ground truth target sequence one by one (using masked inputs for the target sequence) to then compute the loss (Cross Entropy) and thus train the model.

### 4.2 Auto-Regressive Transformer

The second training approach for the DSI Transformer model consists of an "auto-regressive" approach, in which the model only relies on its own past predictions to generate the output tokens used to compute the loss: as for the inference phase described at the start of this section, the model starts from both the query and the "start of sequence" token and generates the final document ID in an "auto-regressive" way during the entirety of its training.

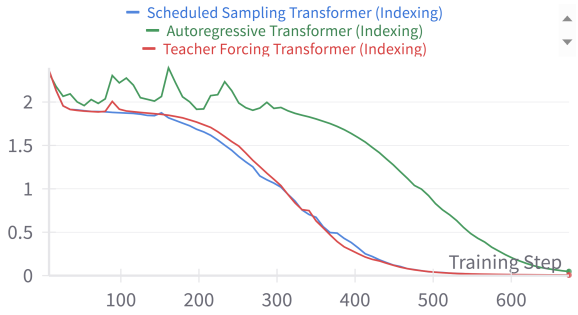
### 4.3 Scheduled Sampling Transformer

The last training approach for the DSI Transformer model, inspired by Mihaylova et al. [2], consists of a mix between the two approaches discussed above: the model starts its training, from the first epoch, by using only the "teacher forcing" approach, but then gradually transitions to a pure "auto-regressive" approach at the end. During this transition, the model learns by either using the ground truth tokens or by generating tokens based on its predictions, with a random "scheduled sampling probability" for each training example, hence a probability of using teacher forcing rather than an auto-regressive approach. This probability decreases linearly at the end of each training epoch, based on a "scheduled sampling probability decay" hyperparameter. For each sample, once the random probability establishes that the next token should be generated by the model using the "auto-regressive" approach, then all the next tokens will be generated using only auto-regression for that sample.

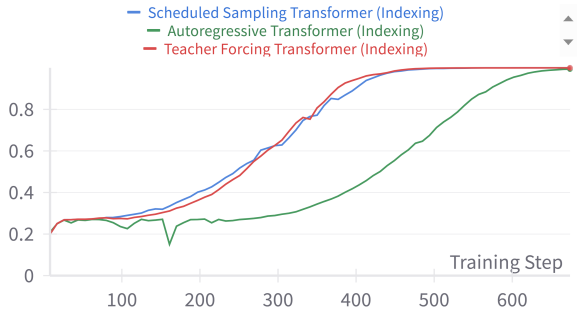
### 4.4 Transformers Training Results

The following graphs (Figures 1, 2, 3 ) show the training and validation loss over epochs for the transformer models trained with the three different approaches, for both the indexing and retrieval task, along with the training and validation accuracy (for the retrieval task).

For both the indexing and the retrieval tasks and for all the three variants, transformers were trained considering a batch size of 1024 and employing early stopping with a maximum of 75 training epochs.

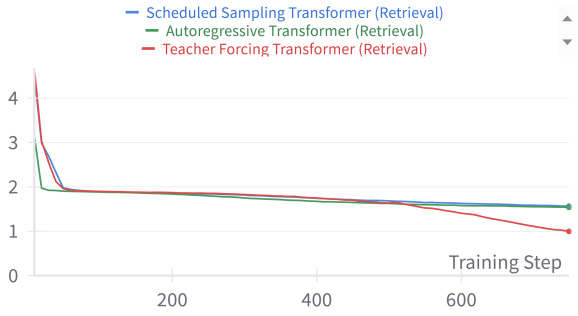



(a) Indexing Train Loss

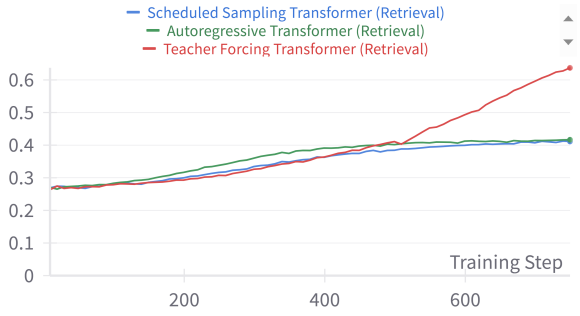



(b) Indexing Train Accuracy

Figure 1: Indexing Phase Train Results

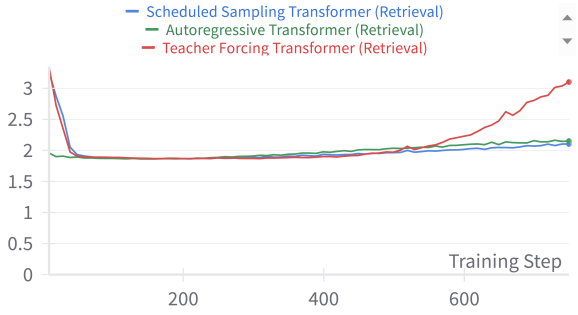



(a) Retrieval Train Loss

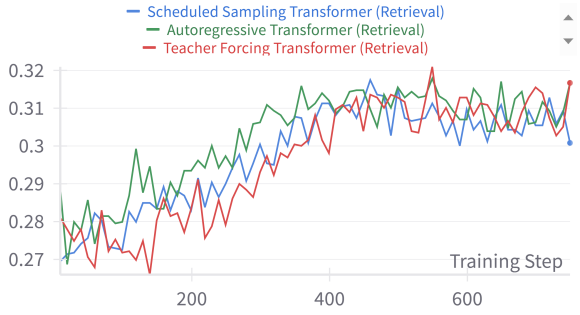



(b) Retrieval Train Accuracy

Figure 2: Retrieval Phase Train Results




(a) Retrieval Validation Loss




(b) Retrieval Validation Accuracy

Figure 3: Retrieval Phase Validation Results

## 5 Results

The following table (Table 1) shows the results of the Transformer models trained using the three aforementioned approaches (see section 4), compared with the three defined baselines (see section 3). The Transformer models have a

simple structure and were trained for a limited number of epochs: these constraints may have influenced the evaluation results, but were forced by the hardware limitations of our training setting.

The "Mean Average Precision" (MAP) metric was computed by considering the number of results outputted by each model out of 10 total outputs with respect to the 10 relevant document IDs of a query (Average Precision, AP), with the final value given by the mean of the AP calculated on 10 different queries (MAP@10).

The "Recall at K" (Recall@K) metric was computed in a similar way by considering the number of relevant outputs out of K total outputs for a query: in order to reduce the variance of this metric, the final value was calculated by considering the mean Recall@1000 over 3 different queries.

Table 1: Models Evaluation Comparison

| **Model** | | **Results** | |
|---|---|---|---|
| **Name** | **Variant** | **MAP@10** | **Recall@1000** |
| TF-IDF[1] | / | 0.89 | 0.98 |
| Word2Vec | / | 0.13 | 0.54 |
| SiameseNetwork | / | 0.15 | 0.76 |
| **Transformer (DSI)** | *Teacher Forcing* | **0.43** | **0.48** |
| **Transformer (DSI)** | *Auto-regressive* | **0.04** | **0.61** |
| **Transformer (DSI)** | *Scheduled Sampling* | **0.34** | **0.52** |

## 6 Conclusion

The conducted study was centered around a Transformer-based DSI framework, faster than traditional "index then retrieve" pipelines, used as a "Neural Inverted Index" for information retrieval tasks. In this study, a novel approach based on "Scheduled Sampling" was compared to traditional "Teacher Forcing" and pure "Autoregressive" Transformers' training settings. The "Scheduled Sampling" approach was employed to try to mitigate "exposure bias": at inference time, if one of the tokens of the output sequence is erroneously predicted, the rest of the tokens predicted for the final sequence may make the final output completely wrong. The results of applying this kind of approach in our Transformer models training can be observed from Table 1: the Transformer model using autoregression during training showed a higher Recall@1000 than the Transformer model using a traditional teacher forcing approach, while the MAP@10 is higher for the latter; the Transformer model using the "Scheduled Sampling" technique seemed to find a balance between the two other approaches, and thus seemed to be more accurate at generating large number of relevant document IDs (top 1000) without presenting a very low accuracy for the generation of the most relevant results (top 10).

Finally, it is worth noting that the setting employed for the DSI model was that of non-semantic document IDs: unlike the original DSI proposal by Tay et al. [1], the IDs of each model in the database was not assigned by clustering documents into a tree-based structure, but was instead assigned at random after shuffling the list of documents from the original MS Marco dataset. This approach was aimed at overcoming the limitation of the original proposal, which made it necessary to assign a new, recomputed semantic ID to all the documents in the database in case of a new document addition. With the proposed approach, in case of new document additions to the database, the first available IDs can easily be assigned to each of the new entries, without needing to retroactively update IDs for the other documents.

## References


[1] Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index, 2022.

[2] Tsvetomila Mihaylova and André F. T. Martins. Scheduled sampling for transformers, 2019.

[3] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2018.


[1] No machine learning used for the TF-IDF model