# Neural Inverted Index for Fast and Effective Information Retrieval

DIPARTIMENTO DI INGEGNERIA INFORMATICA
AUTOMATICA E GESTIONALE ANTONIO RUBERTI

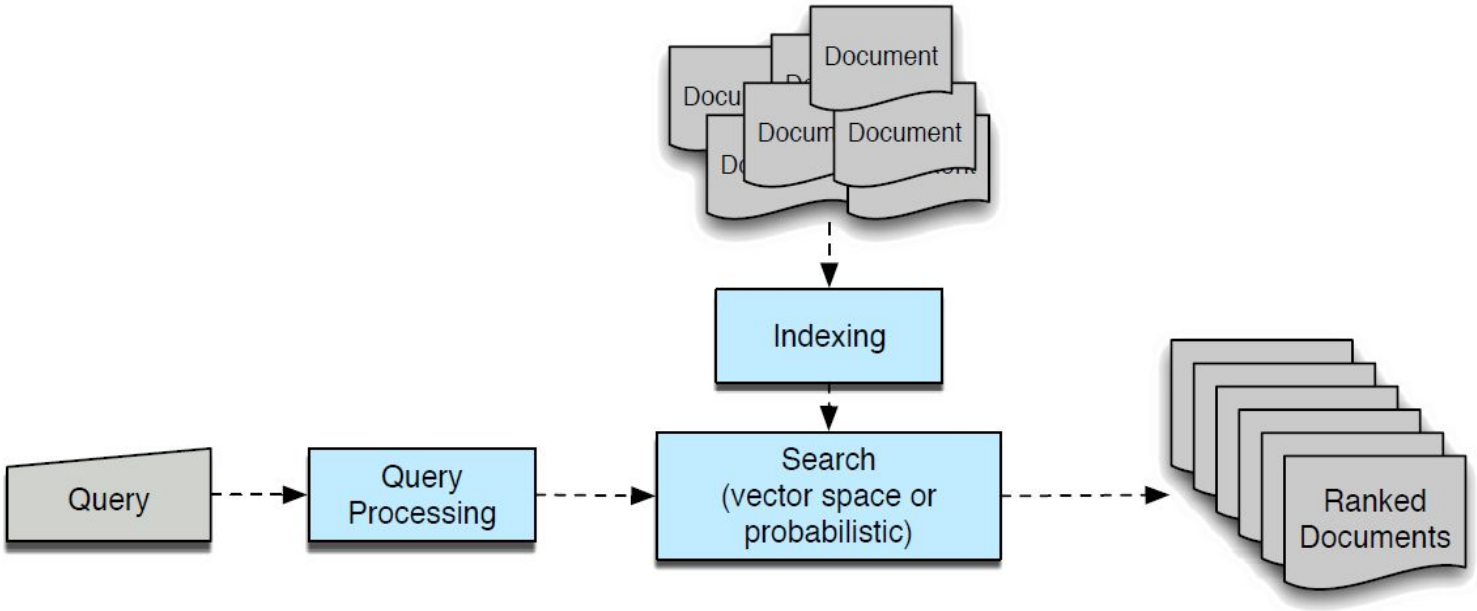

SAPIENZA
UNIVERSITÀ DI ROMA

**Master in Artificial Intelligence and Robotics**

**Course** Deep Learning
**Professor** Fabrizio Silvestri
**Students** Luigi Gallo (1895146),
Bonifacio Marco Francomano (1883955)

# Introduction to the task

The **goal** of the work showed in this presentation was to develop a method for **obtaining** a list of **relevant documents given a query**, incorporating the entire pipeline into a deep learning model.



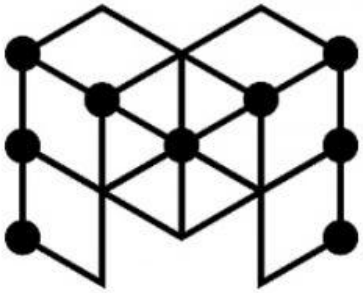
# Introduction to the task (2)

Due to the fact that the task is very challenging and the computational resources are limited, in this work we try to examine **several approaches**, in order to understand which are the main ways to explore in order to get significant results in a more advanced environment of training.

For this reason, we have tried different approaches on the **preprocessing** side, on the **data representation** side, and on the modeling of how the data is processed (**architecture** and **inference**).

# Dataset Creation

The preliminary step involves the creation of two dictionaries (**queries** and **documents**). Iterating over all MS MARCO queries (and documents), the 'raw' text is stored in both dictionaries (using query and document IDs as keys), along with a list of relevant documents for each query.

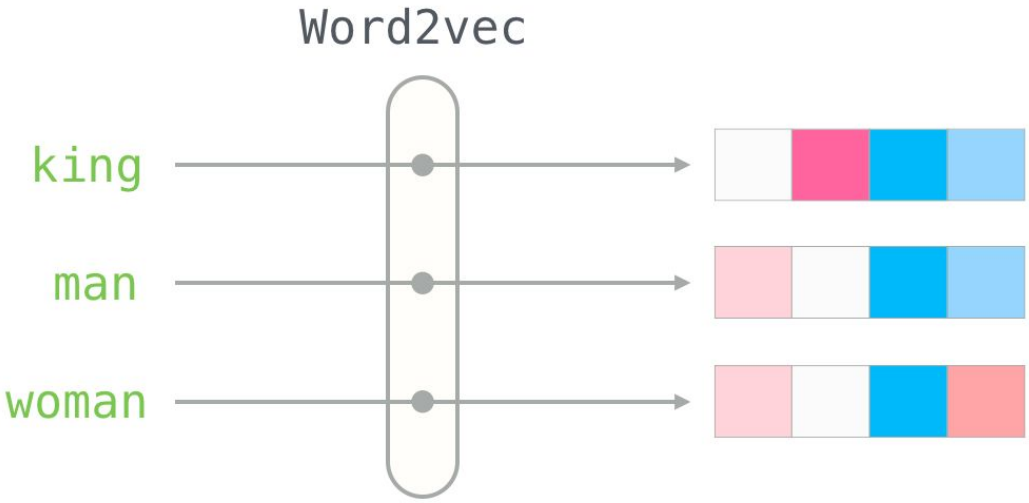


MS MARCO

Microsoft MAchine Reading COmprehension Dataset

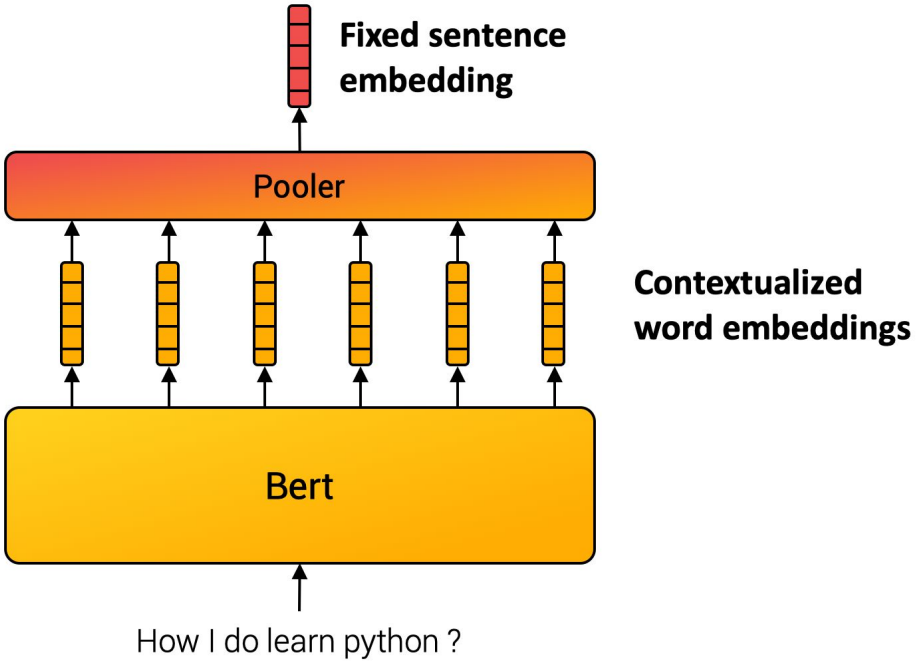
# Dataset Creation (2): Word Embeddings and pre-processing

Since the first baseline we chose to build is a model that works directly on the content of documents and queries, **word2vec** was used to build the **embeddings** of the preprocessed samples (no stop-words, punctuation and lemmatized).



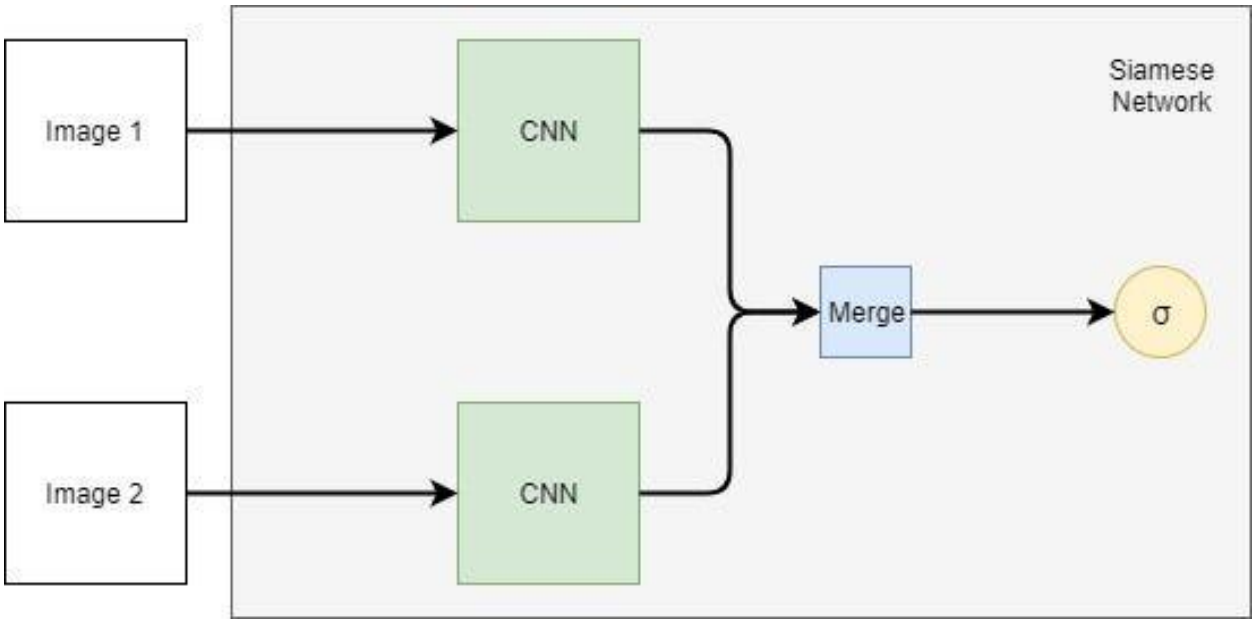
# Dataset Creation (3): Document and Query Embeddings

For the creation of doc and query **embeddings**, we tried two different approaches: *average* and *first L tokens* embeddings.



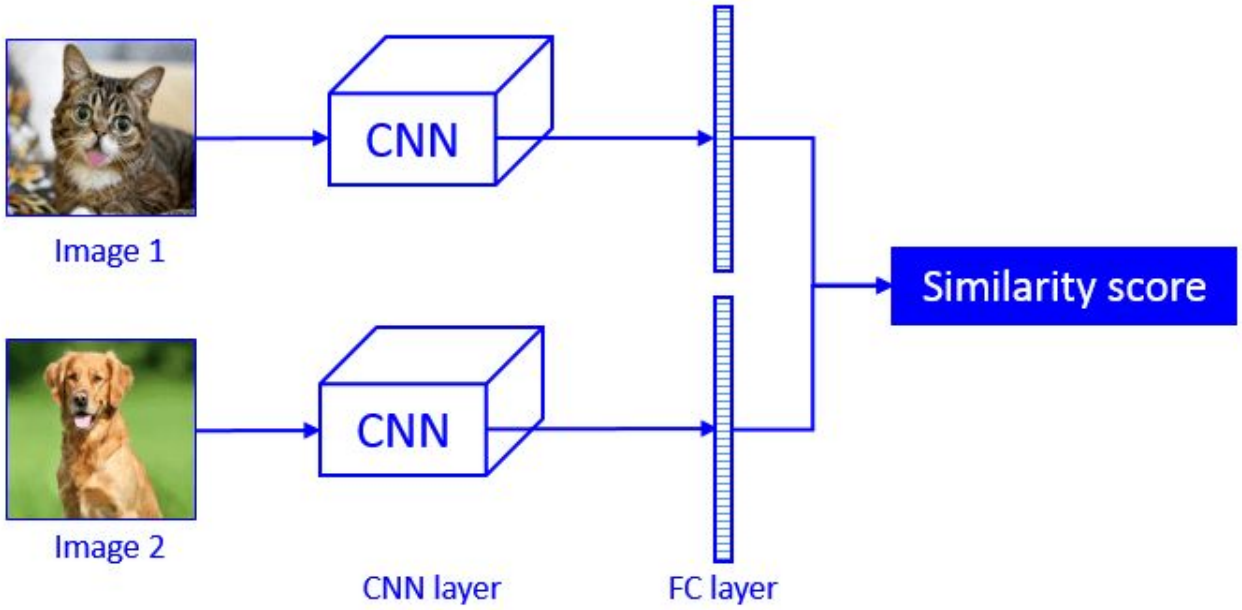
# First Baseline: Simple siamese (classifier)

It processes the embeddings of queries and documents, outputting the **probability** of their **correlation** by simply concatenating the two feature vectors, and passing the final vector into a fully connected layer followed by a sigmoid.



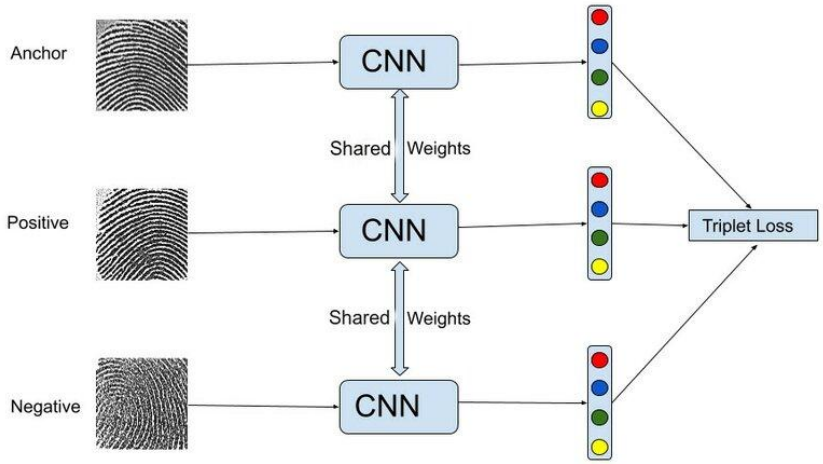
# Second Baseline: Similarity score

- Sees directly the embeddings obtained by the *first L tokens* of the documents and the queries, in order to compute **attention** scores for building the two feature vectors.
- It generates a **relevance** score between them, this time by the use of **cosine similarity**.



# Third Baseline: Contrastive Learning

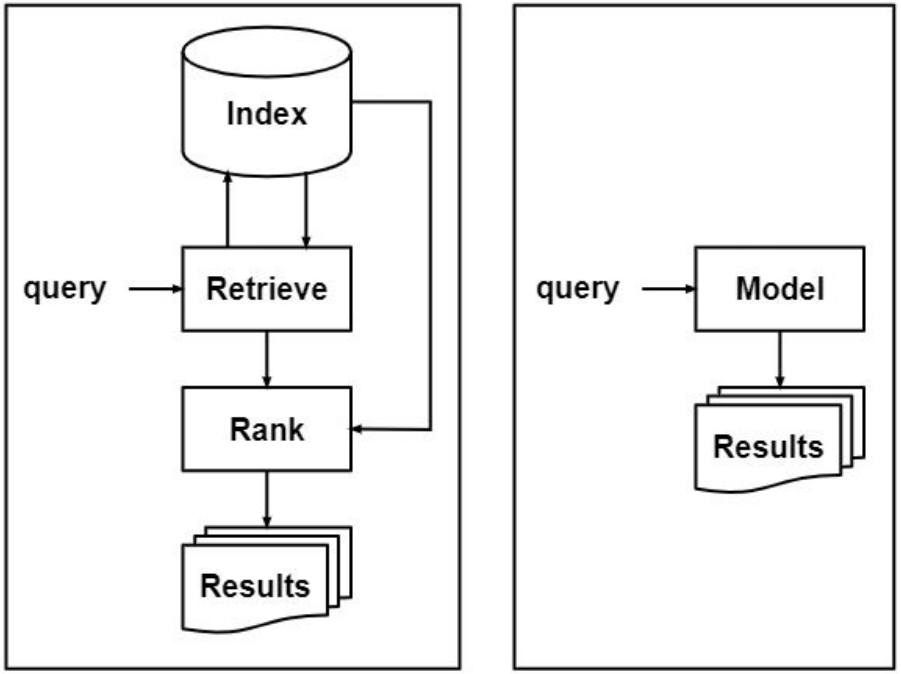
The final baseline, “Siamese Triplet”, was trained using a **Triplet Margin Loss**, by providing positive and negative examples corresponding to the queries.




$$Loss = \sum_{i=1}^{N} \left[ \| f_i^a - f_i^p \|_2^2 - \| f_i^a - f_i^n \|_2^2 + \alpha \right]_+$$

# DSI Transformer based model

This approach integrates the **indexing** and **retrieval** processes into a singular, cohesive **Transformer** language model, diverging from conventional bifurcated IR system designs.

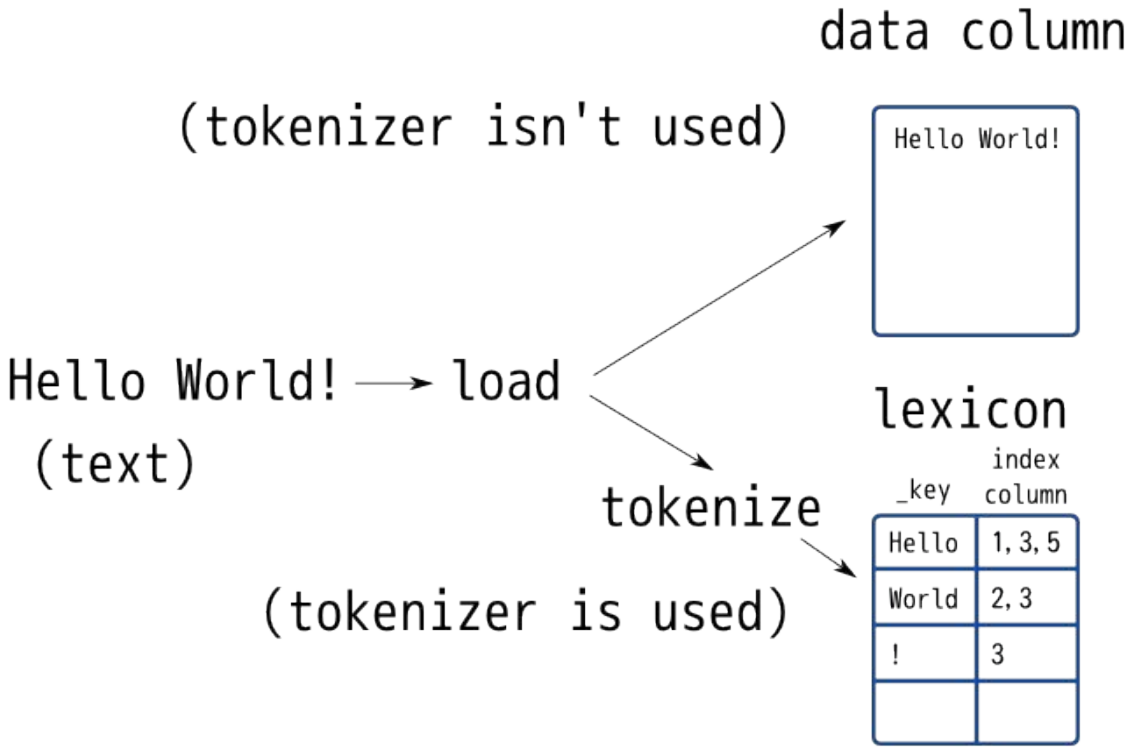



**(a)** Retrieve-then-rank **(b)** Unified retrieve-and-rank

# Preprocessing

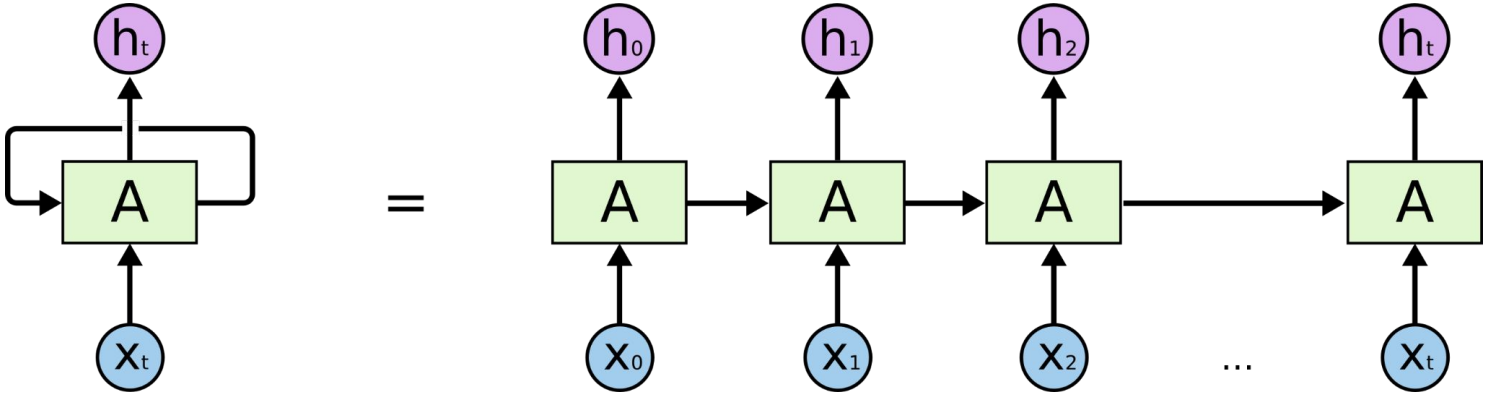
**T5-small** tokenizer

- First L=32 tokens for representing the documents
- First L=9 for representing the queries



# DSI Transformer based model (2): how it works

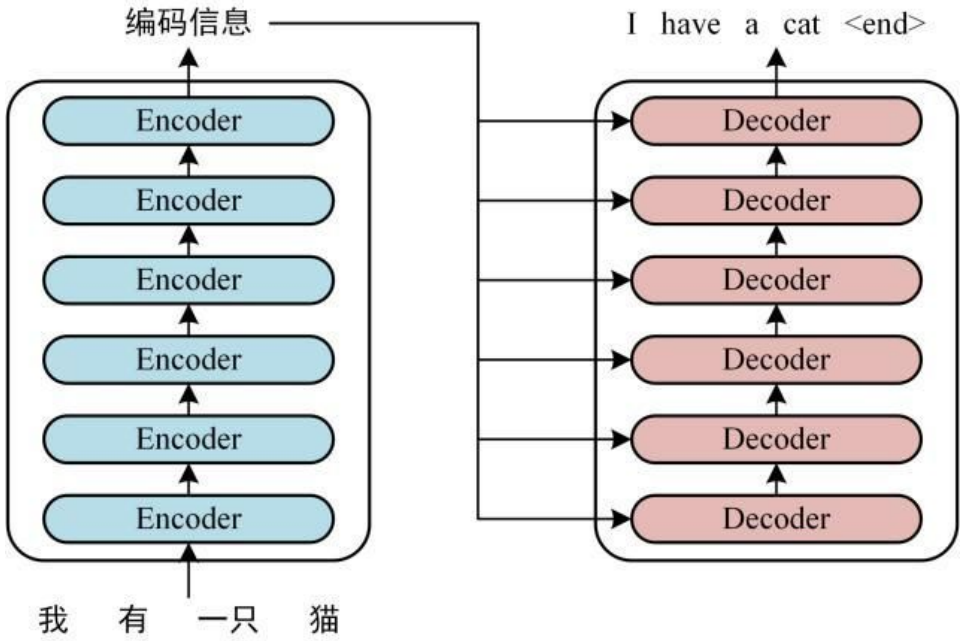
Our project embarks on constructing a **unified** model that acts as a **sequence to sequence** architecture, which takes as input a query ’q’, and generated in a auto-regressive way, a relevant docid for that query.



# DSI Transformer based model (3): Architecture and Hyper-parameters optimization

- Very **lightweight** sequence-to-sequence model
- **Two** different **embedding layers**, one for the input sequence and one for the target sequence (two different tokenizer)
- **3** encoder-decoder **transformer layers**
- Final **fully connected layer**.

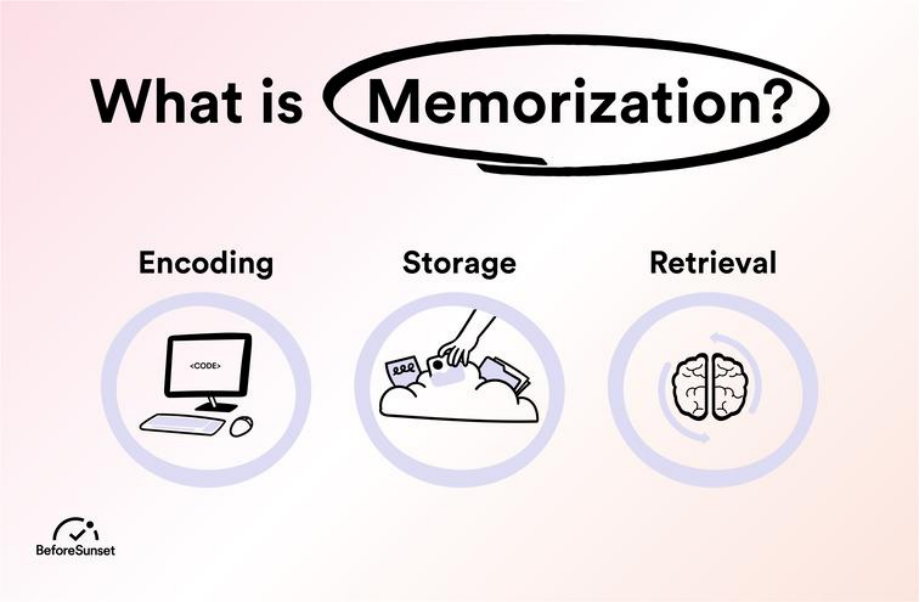
| |
|---|
| 0.2 transformer dropout |
| 0.1 pos.encoding dropout |
| 120 embedding size |
| 120 feedforward size |
| 0.0005 LR (Adam) |



# DSI Transformer based model (4): Training strategy

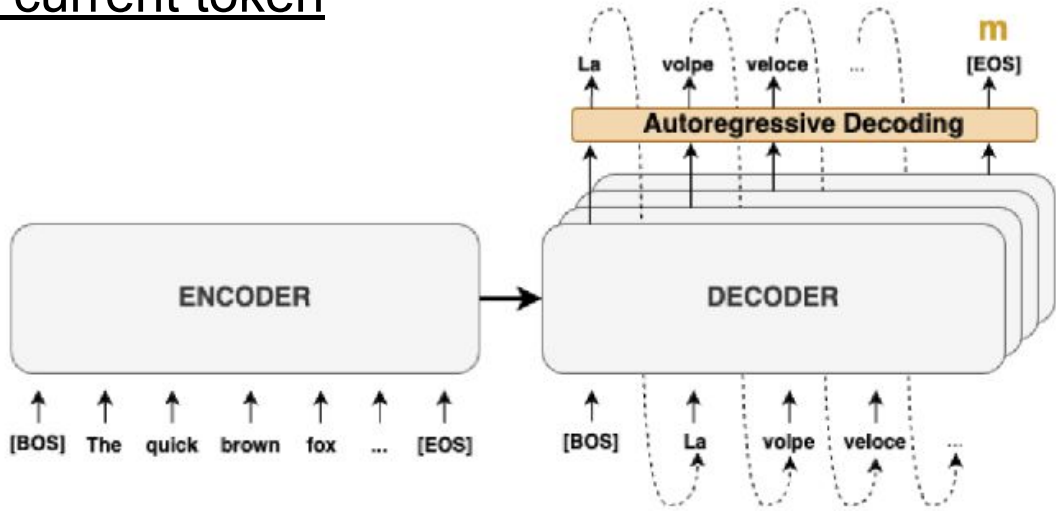
The **training** setup was split in two main sections:

- **Indexing**: the model has to acquire the documents knowledge inside its parameters
- **Retrieval**: the model must generate starting only from the special start token the entire target sequence given a query.



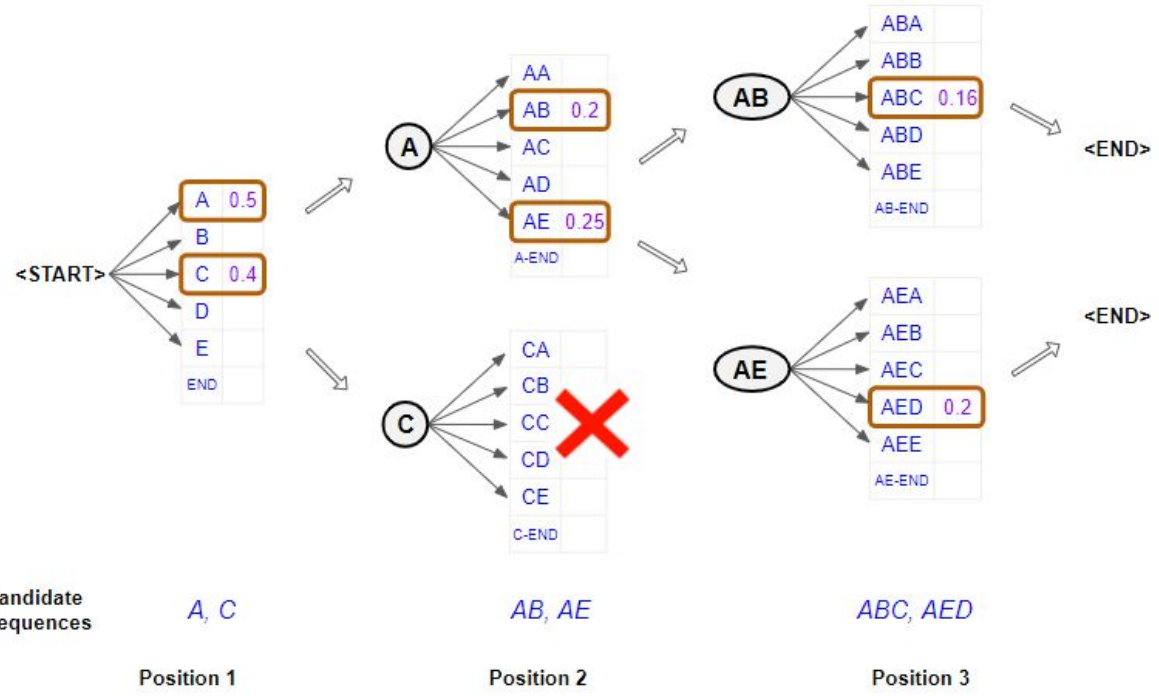
# DSI Transformer based model (5): Training strategy

- We tried to modify the training step in order to use **teacher forcing** with a certain **probability** (initialized to 1), and this probability **decreases** every epoch. When the teacher forcing is not used, we enter in the **autoregressive** setup
- The final idea was to modify the autoregressive modality in order to feed the decoder with all the generated sequence up to the current token



# Inference: Constrained Beam Search

- We **constrained** the model to generate only **existent** docids, by using a special data-structure called "Trie".
- In doing so, only the **top k** token sequences with higher **probabilities** are **kept**, in a "beam-search fashion".



# Results on the test set

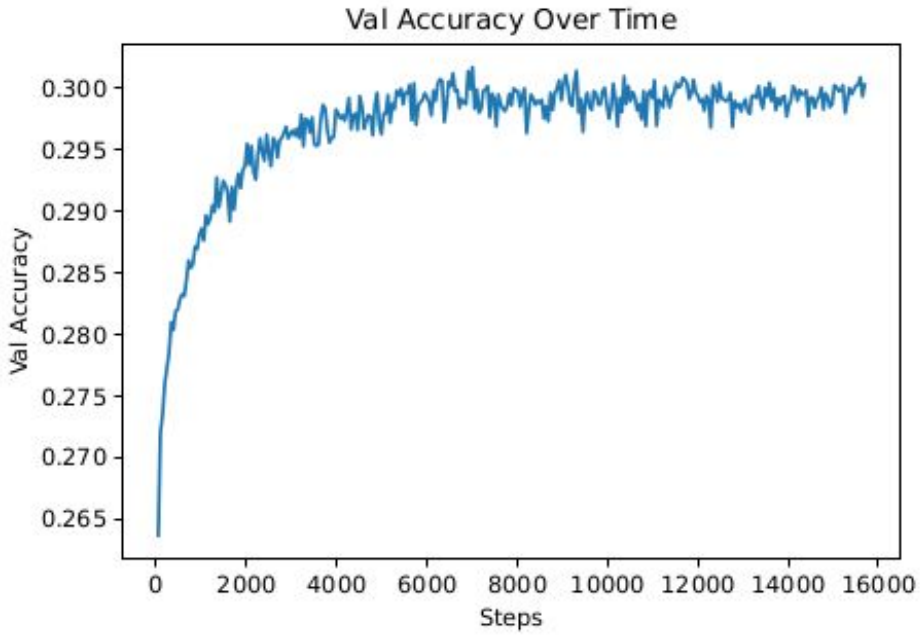
- Due to the **limits** of usage imposed by Google Colab (RAM saturation and GPU usage), we couldn't train for more than an hour
- For sure with more **powerful** computational **resources** we can achieve **higher results**, but the table below shows that the setup we have adopted begins to show promising results.

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 | / | / |

# Results (2) : validation accuracy

The **accuracy** on the **validation** set, shows a consistent **increase**. This implies that the model's predictions are more often correct, which suggest that the model is increasingly better at identifying the right document identifiers over time.



# Conclusions

- analyze the task of integrating indexing and retrieval processes into a singular cohesive transformer language model
- application of various training strategies, (auto-regressive and teacher-forcing methods)
- improvement in retrieval effectiveness with respect to the classic self-supervised models and the classic sequence to sequence models trained to predict a single missing token.
- challenges : the limitations in computational resources restricted the extent of training, suggesting that further improvements could be achieved with more powerful computational setups.

In conclusion, this unified approach opens the way for more efficient, accurate, and user-friendly information retrieval systems.