Evaluation of the performance of three *Sentence Transformers* text embedders - a case study for SenTrEv

Astra Clelia Bertelli[1]

Abstract: As of december 2024, more than 10,000 models are available within the *sentence-transformers* python library: with the growing number of Retrieval Augmented Generation applications, it is more than necessary to employ an easy-to-use evaluation method to find the best text embedder for retrieval purposes. Sentence Transformers Evaluator (SenTrEv) is a python package that was developed to provide user with a simple evaluation of dense retrieval performance of Sentence Transformers text embedding models on PDF data. In this case study, we present the potential application of SenTrEv in the evaluation of three popular *sentence-transformers* text embedders on a small PDF dataset, reporting and interpreting the statistics related to retrieval success rate, time performance, mean reciprocal ranking and carbon emissions.

1. Introduction

The growth of Artificial Intelligence (AI) from late 2022 has benefited numerous fields, ranging from chat models to computer vision to the generation of audiovisual material.

A field that has been growing, since AI-based text generation has been given special attention from the generalist public, is the one of Retrieval Augmented Generation (RAG). The idea behind RAG is simple but powerful:

- it starts from some text-based material (being it PDFs, CSVs, HTML websites, Markdown documents...)
- the text is chunked, i.e. subdivided into smaller pieces to make it more "digestible"
- a model (called text embedder, encoder or vectorizer, and those expressions will be used interchangeably in this study) takes care of generating a multi-dimensional vector representation of the text
- the vector representations are loaded into a vector database (there are many popular services that offer this kind of data storage)
- when the chat model is prompted with a query, this same query is firstly embedded into a vector and then used to retrieve similar chunks of texts from the vector database: these chunks of text will serve as context that may enable the model to produce an informed answer to the user's prompt

This approach, although sometimes much more complicated than the core here reported, has proven to substantially enhance AI models performance.

Nevertheless, RAG comes with an important hassle: one cannot assess the goodness of a text embedder until it tests it in action, which could lead to two main consequences -either the model is too small for the pipeline to which it is applied, resulting in low-quality embeddings and retrieval calls, or the model is too big, ending up in a waste of computational and (potentially) financial resources. It can thus be concluded that an evaluation framework for embedding models should be employed to establish their fitness for the task before they are deployed in any test or production environment.

A potential solution to evaluate the dense retrieval performance of models is SenTrEv (**Sen**tence **Tr**ansformers **Ev**aluator; Bertelli, 2024).

SenTrEv is a Python package offering a simple retrieval evaluation frameworks that takes into account dense retrieval accuracy, time performance and carbon emission tracking. Its applicability concerns PDF data and extends to all 10,000+ embedders available through the popular `sentence-transformers` Python library (https://sbert.net, last visited 12-04-2024), leveraging `Qdrant` (https://qdrant.tech, last visited 12-04-2024) as vector storage.

In this case study, we showcase SenTrEv evaluation potential by comparing three popular `sentence-transformers` based embedding models on a small PDF dataset.

2. Materials and Methods

2.1 Embedding models

As already mentioned in the introduction, SenTrEv was employed in the evaluation process of three popular `sentence-transformers` encoders: `all-MiniLM-L12-v2`, `all-mpnet-base-v2` and `LaBSE`, whose technical details are reported in Table 1.

Model	Base Model	Number of Parameters	Embedding vector dimensions	Reference
all-MiniLM-L12-v2	MiniLM-L12-H384-uncased by Microsoft	1B	384	https://huggingface.co/sentence-transformers/all-MiniLM-L12-v2; **Paper**: Wang et al., 2020
all-mpnet-base-v2	mpnet-base by Microsoft	1B	768	**HuggingFace**: https://huggingface.co/sentence-transformers/all-mpnet-base-v2; **Paper**: Song et al., 2020
LaBSE	LaBSE by Google	17B English sentence	768	**HuggingFace**: https://huggingface.co/sentence-

Model	Base Model	Number of Parameters	Embedding vector dimensions	Reference
		pairs + 6B Multi-Lingual Sentence Pairs		transformers/LaBSE, **Paper**: Feng et al., 2020

Table 1: Technical details and references for the evaluated embedding models

2.2 Data and preparation

The evaluation data employed were the PDF version of two landmark papers in the field of AI: *Attention is all you need* (Vaswani et al., 2017) and *Generative Adversarial Networks* (Goodfellow et al., 2014).

The PDFs text was extracted via a PyPDF (https://pypdf.readthedocs.io/en/stable/; last visited 12-04-2024) wrapper offered by LangChain (https://langchain.com; last visited 12-04-2024).

The extracted text was chunked in sizes of 500, 1000 and 1500 characters by LangChain text splitter.

These chunks were uploaded to a local Qdrant-operated vector database via `qdrant-client`, Qdrant's python bindings (https://pypi.org/project/qdrant-client/; last visited 12-04-2024).

The PDF preprocessing and uploading is integrated in SenTrEv framework.

2.3 Evaluation workflow

SenTrEv applies a very simple evaluation workflow:

1. After the PDF text extraction and chunking (cfr. *supra*) phase, the chunks are reduced according to a (optionally) user-defined percentage (default is 25%), which is randomly extracted at any point of each chunk.
2. The reduced chunks are mapped to their original ones in a dictionary
3. Each model encodes the original chunks and uploads the vectors to the Qdrant vector storage
4. The reduced chunks are then used as queries for dense retrieval
5. Starting from retrieval results, accuracy, time and carbon emissions statistics are calculated and plotted.

See Fig 1 for a visualization of the workflow

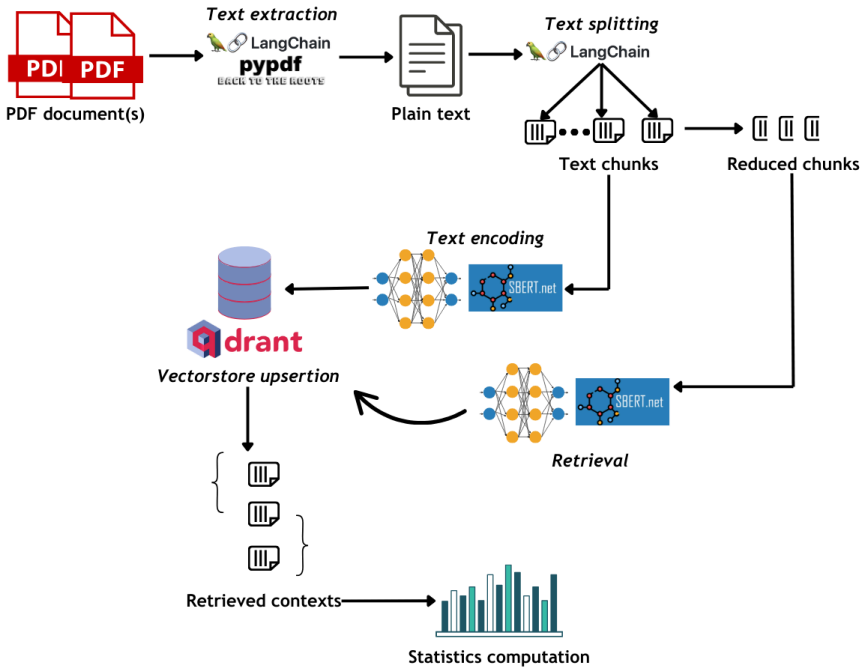

Fig. 1: Evaluation workflow for SenTrEv

In this study, the text percentages for chunk reduction were set to 40, 60 and 80%.

Additionally, the distance metrics were tweaked, exploring all available ones for Qdrant, i.e. cosine, euclidean, Manhattan and dot product distance.

Accounting also for the different chunking sizes (cfr. *supra*), this study presents 36 total test runs, each with a different combination of the aforementioned parameters. We then averaged across all these tests too see which model performed better and what factors mostly influenced their performance.

All the tests were conducted with SenTrEv v0.1.0 on Python 3.11.9, exploiting a Windows v10.0.22631.4460 machine. Models were loaded on GPU for faster inference (GPU hardware: NVDIA RTX4050, 6GB ggr6-sdRAM; CUDA v12.3).

3. Results and discussion

The metrics used to evaluate performance were:

- **Success rate**: defined as the number retrieval operation in which the correct context was retrieved ranking top among all the retrieved contexts, out of the total retrieval operations:

  $SR = \frac{Ncorrect}{Ntot}$ (eq.1)

- **Mean Reciprocal Ranking (MRR)**: MRR defines how high in ranking the correct context is placed among the retrieved results. MRR@10 was used, meaning that for each retrieval operation 10 items were returned and an evaluation was carried out for the ranking of the correct context,

which was then normalized between 0 and 1 (already implemented in SenTrEv). An MRR of 1 means that the correct context was ranked first, whereas an MRR of 0 means that it wasn't retrieved. MRR is calculated with the following general equation:

$MRR = \frac{ranking + Nretrieved - 1}{Nretrieved}$ (eq.2)

When the correct context is not retrieved, MRR is automatically set to 0. MRR is calculated for each retrieval operation, then the average and standard deviation are calculated and reported.

- **Time performance**: for each retrieval operation the time performance in seconds is calculated: the average and standard deviation are then reported.
- **Carbon emissions**: Carbon emissions are calculated in gCO2eq (grams of CO2 equivalent) through the Python library `codecarbon` (https://codecarbon.io/; last visited 12-04-2024) and were evaluated for the Austrian region. They are reported for the global computational load of all the retrieval operations.

3.1 Embedders performance evaluation

The three models were evaluated according to their accuracy, time and carbon emissions metrics.

As you can see from Fig 2A, `LaBSE` proved to be the best model for what concerns success rate, yielding an average of 74.3% correct retrieval calls. In this case, `all-MiniLM-L12-v2` proved better (although not much better, given the 4.2% advantage) than `all-mpnet-base-v2`, despite using vectors with a smaller number of dimensions (384 against 768).

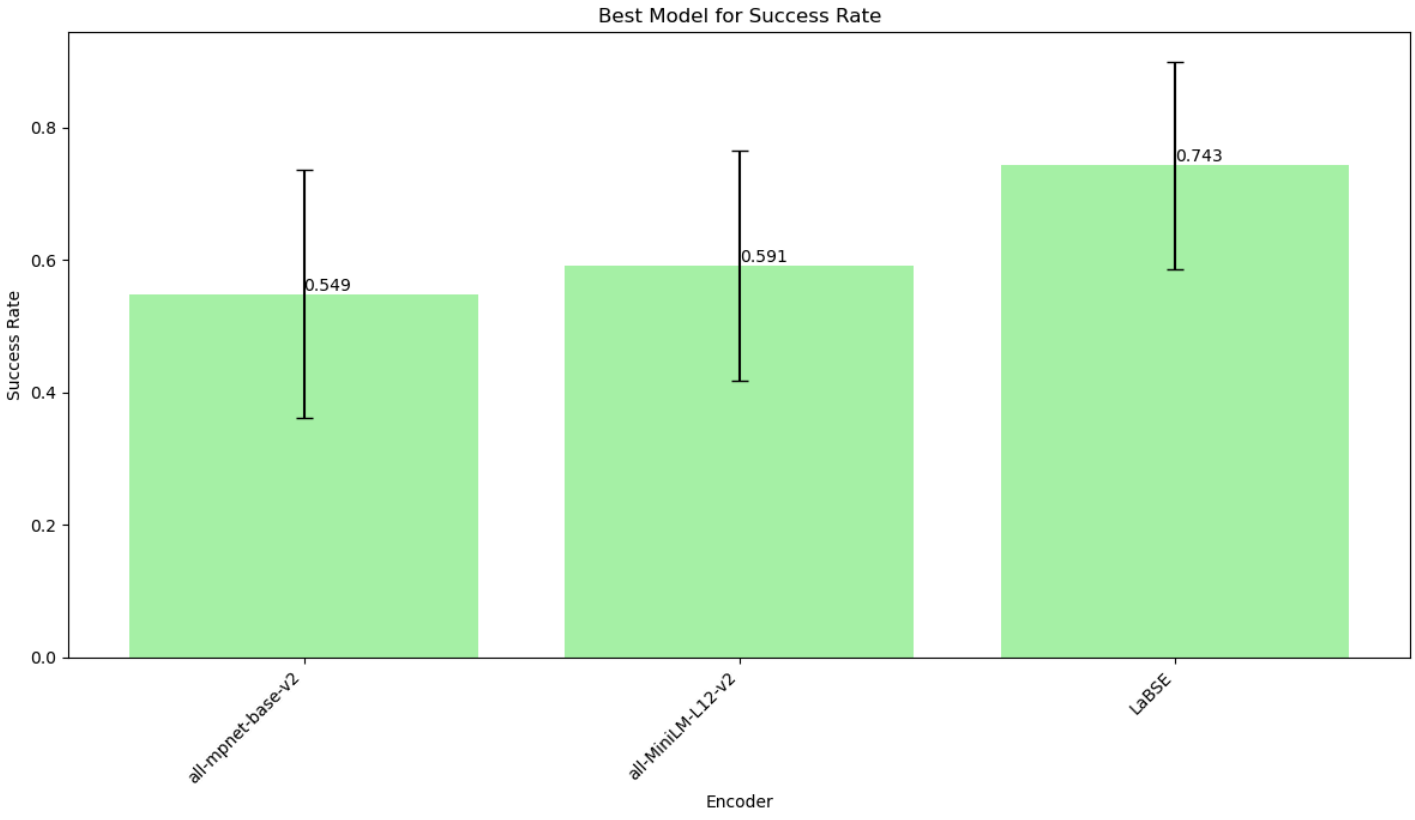

Fig.2A: average success rate for the three text embedders

Fig 2B, instead, displays the performance of the embedders in terms of average MRR@10. In this sense, the three models do not show a highly significant difference in performance, with `LaBSE` leading the ranking at 91.5%, followed by `all-MiniLM-L12-v2` with 88.7% (-2.8%) and by `all-mpnet-base-v2` with 85.6% (-5.9%). This means that, on average, the correct context was always retrieved among the first 3 positions by every model, and may hint at the necessity of taking into account more than the top one retrieval result when building a RAG application.

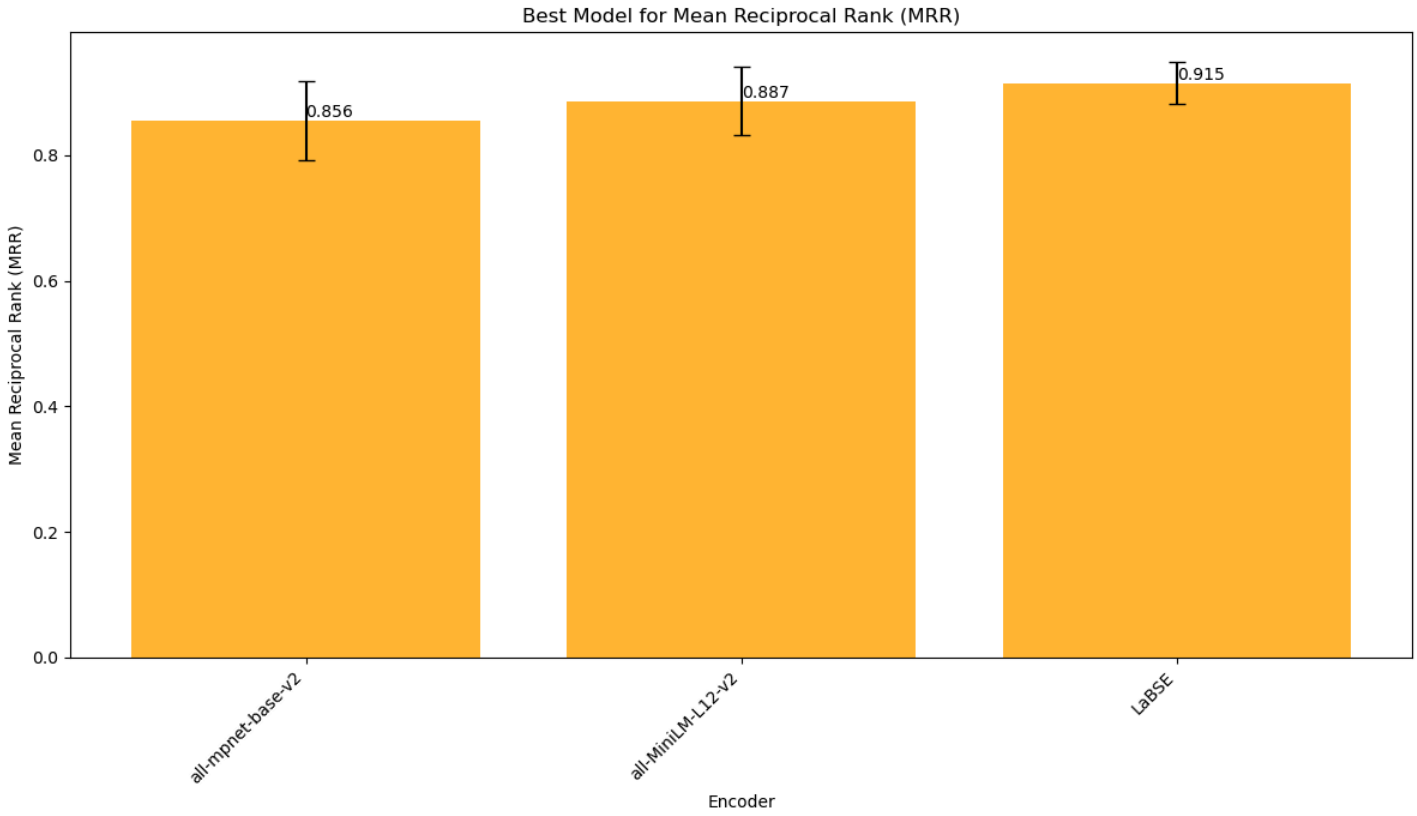

Fig.2B: average MRR@10 for the three text embedders

For what concerns time performance (Fig 2C), `LaBSE` and `all-MiniLM-L12-v2` rank both in the first position with an average of 0.038s per retrieval operation, whereas `all-mpnet-base-v2` proved a little slower with an average of 0.041s. Although the time difference in this case seems trivial, it is crucial to take it into consideration: a 0.003s difference, on 1 million retrieval call, will account for a 50 minutes delay.

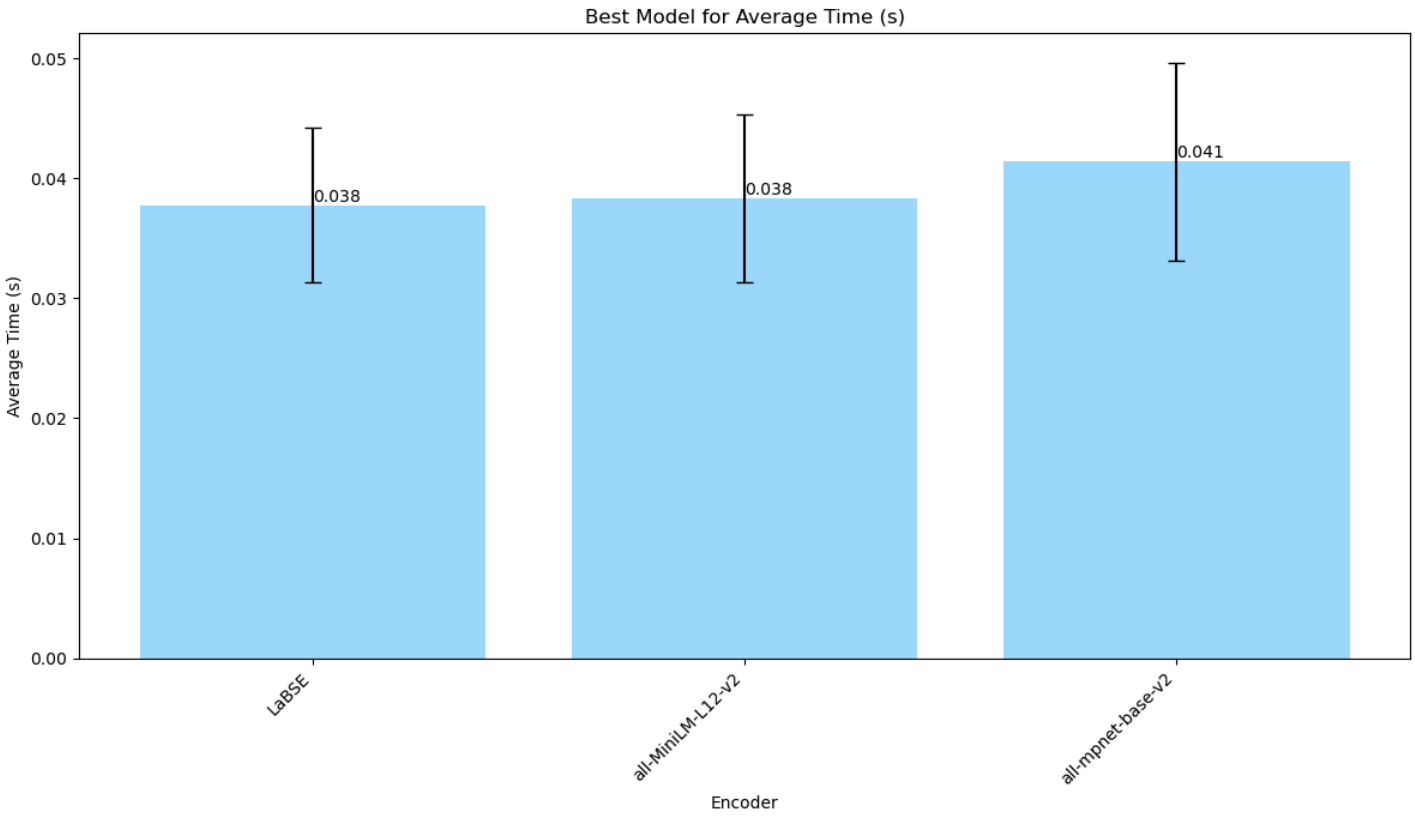

Fig.2C: Average time performance for the three text embedders

Finally (Fig 2D), `all-mpnet-base-v2` ranked as the model with the least carbon emissions, with an average of 0.173 gCO2eq against `all-MiniLM-L12-v2`, which reaches almost double the emissions with 0.320 gCO2eq, and `LaBSE`, that shows an average of 0.473 gCO2eq. If the carbon emissions performance of `LaBSE` doesn't come unexpected (it's by far the largest model among the three), it is interesting to see how `all-mpnet-base-v2` manages to produce less carbon emissions than `all-MiniLM-L12-v2`, despite being 3.3 times its size (438 MB vs 134MB in terms of occupied memory space).

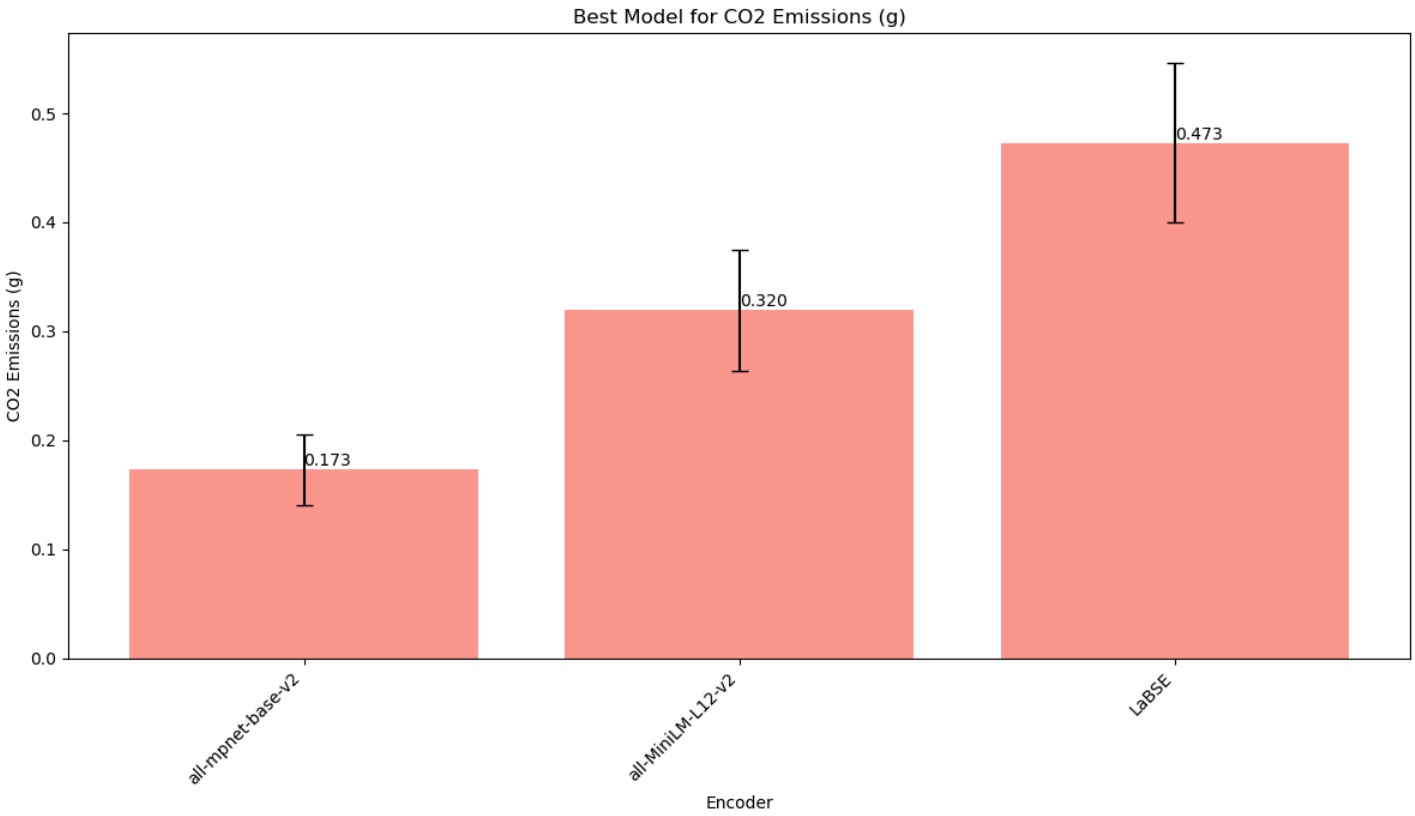

Fig.2D: Average carbon emissions in gCO2eq for the three text embedders

3.2 Factors influencing perfomance

The influence of the three parameters that were modified in the test replicates, i.e. chunking size, text percentage and distance metrics, were taken into account in the combined analysis whose results are shown in Fig. 3A-C.

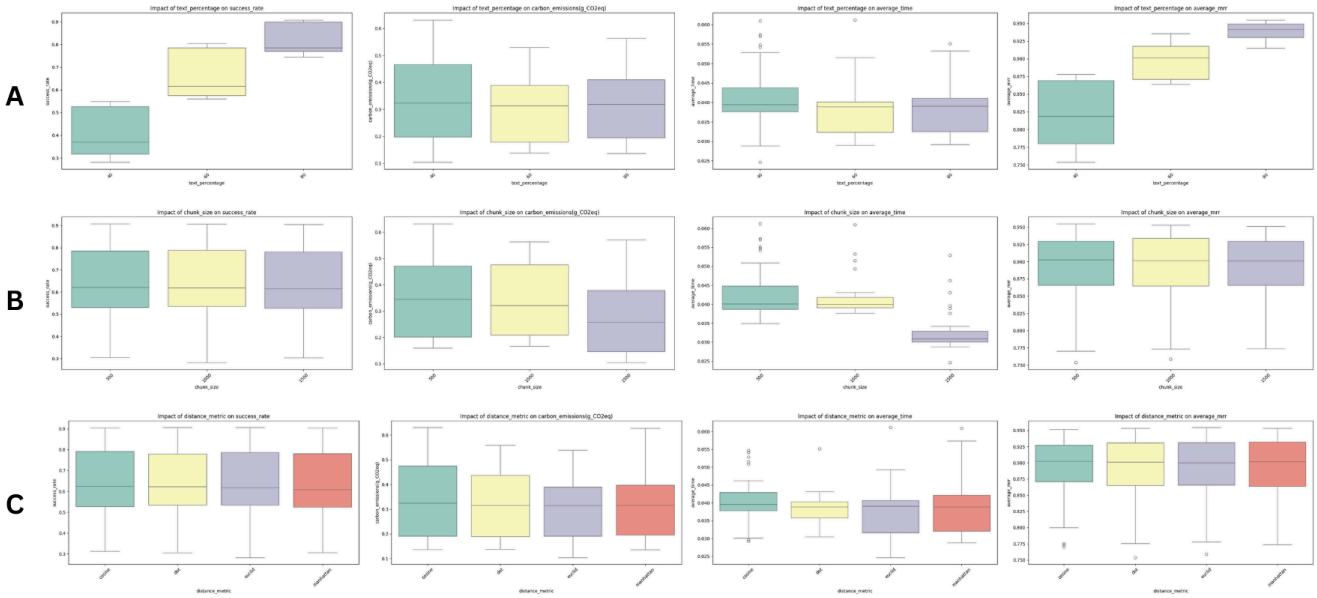

Fig. 3A-C: Factors influencing model performances: **A** is chunking size, **B** is text percentage and **C** is distance metrics

It can be observed that:

- **Chunking size** appears to have no effect whatsoever on success rate and on MRR, whereas it shows a negative correlation with average retrieval time and carbon emissions. The first can be explained by the fact that the larger the chunking size, the fewer the chunks: this means that the vector databases contains fewer data points and so retrieval goes faster. In the case of carbon emissions, the correlation is due to the fact that the fewer the chunks, the fewer the retrieval operations: since each operation has a more or less fixed cost, if we have less operation we will be having less carbon emissions. This is nevertheless to be adjusted by the larger chunking size, which may take up more energy (and thus more CO2eq emissions) for encoding the text chunks.
- **Text percentage** appears to have a strongly postive effect on both success rate and MRR. This is explained by the fact that the larger the reduced chunk is in comparison to its original one, the more similar it will be: since the semantic search is similarity-based, it can be concluded that a higher similarity between the query and the correct context makes it easier for the retriever to operate a correct retrieval. Text percentage doesn't appear to significantly influence average retrieval time and carbon emissions.
- **Distance metrics**: distance metrics, in these tests, do not appear to have an influence over any of the evaluation metrics that were taken into account. This does not mean that they shouldn't be explored: as much as they didn't do any difference for this small test, they could make a great difference with larger datasets.

4. Conclusion

In this case study, the evaluation of three sentence-transformers models was presented as a benchmark for SenTrEv evaluation.

SenTrEv proved to be providing a solid, yet simple and effective evaluation workflow, with intuitive and easily interpretable metrics. Nevertheless, the application of retrieval in RAG are countless and there are numerous way in which one could improve the performance of their retriever, adding steps to the RAG process or considering expanding it to other types of retrieval, such as sparse or hybrid search. SenTrEv, moreover, produces only a few of the metrics that can be used, but it does it to make the evaluation of the retrieval model accessible and reliable at-a-glance, without the need for complex and obscure workflows.

SenTrEv, thus, while still lacking the complexity that many workflows could potentially present, builds on interpretability and openness as its core values, making retrieval evaluation simple but reliable.

5. Code and data availability statement

All code and data are contained in the dedicated GitHub repository:
https://github.com/AstraBert/SenTrEv-case-study

6. References

- Wang W., et al., *MiniLMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers*, 2020, arXiv, doi:10.48550/arXiv.2012.15828
- Song K., et al., *MPNet: Masked and Permuted Pre-training for Language Understanding*, 2020, arXiv, doi:10.48550/arXiv.2004.09297
- Feng F., et al., *Language-agnostic BERT Sentence Embedding*, 2020, arXiv, doi:10.48550/arXiv.2007.01852
- Bertelli A. C., *SenTrEv - Simple customizable evaluation for text retrieval performance of Sentence Transformers embedders on PDFs (v0.0.0)*, Zenodo, doi:10.5281/zenodo.14212650

1. Deparment of Biology and Biotechnology, University of Pavia