FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

David Burian

Document embedding using Transformers

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	Jindřich, Libovický Mgr. Ph.D.
Study programme:	Computer Science
Study branch:	Artificial Inteligence

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In .... date ....

Author's signature

Dedication.

Title: Document embedding using Transformers

Author: David Burian

Institute: Institute of Formal and Applied Linguistics

Supervisor: Jindřich, Libovický Mgr. Ph.D., Institute of Formal and Applied Linguistics

Abstract: Abstract.

Keywords: text embedding document embedding transformers document classification document similarity

Contents

Introduction

- what we are aiming to do
- why is it important/useful — where are embeddings used
- long documents — why?
- transformers — why?

1. Document representation

In this chapter we define the properties of a document representation we strive for. The following definitions create a foundation both for comparing models and choosing evaluation tasks.

Before enumerating individual properties, we formally define, what we mean by document representation. Here we follow the concept of embedding, so often encountered in deep learning; representation of an object is understood as a vector of floating point numbers.

Definition 1. *Document representation (or embedding) is a mapping from continuous piece of text to a dense vector of floating point numbers.*

The properties align with the goal of this work; which is *to create semantic embeddings of long texts*. While this definition is concise, it lacks the exactness which will be needed, once we we make decisions about our model.

The properties are split into two sets: *internal* and *external*. Internal properties inspect how the representation is computed. External properties examine the representation itself and how it behaves and what information it contains.

Both types of definitions are difficult to test. As is generally known neural networks are black boxes, whose computations are often not understood completely. This makes studying both internal and external properties challenging and sometimes even impossible. Consequently we try to be cautious with the properties' definitions and keep the testability of given definition in mind.

1.1 Internal properties

1.1.1 Long documents

As our goal is to embed long pieces of texts, it is natural to demand that the model used to generate the embedding is able to process whole documents. Since the meaning of a longer text can be distributed unevenly throughout it, the model should see the entire text before generating an embedding. This gives the model the chance to decide what is important and what can be left out.

Property 1 (long-inputs)**.** *Document representation should be able to capture content from the entire document, not just its part.*

Notice that we avoided specifying an exact threshold of input's length. While this would make the property more testable, we believe that no threshold is truly sufficient.

1.1.2 Context and text structure

It has been shown that natural language tends to be ambiguous, especially with shorter contexts TODO: citation. To capture meaning, the model producing the embedding should contextualize every piece of text found on its input. In other words it should have the ability to compare representations of various words and/or sentences and/or paragraphs with each other. This gives us a chance that the embedding produced by such model will contain the overall meaning of the document rather than meaning of its individual parts.

Having

Besides capturing the whole document, the representations should contextualize the information found.

Additionally it is important to view the pieces of input always in the context of the whole document. This is because longer context helps with the ambiguity of natural languages.

2. Related Work

- what has been done
- approaches to our problem
- mention models we will use

2.1 Models

In this section we describe a set of benchmark models our model will be compared to. All of the benchmark models are able to map a continuous piece of text to a dense vector representation. This was a requirement as the aim of the evaluation is to compare different text embeddings. Otherwise we aimed to select a variety of models with different architectures, learning algorithms and learning tasks.

2.1.1 Paragraph Vector

Paragraph Vector (also known as Doc2Vec) introduced in Le and Mikolov [2014], combines slightly altered DBOW and DM architectures previously used by Word2Vec in Mikolov et al. [2013]. As seen in Figure 2.1 the model's versions of DBOW and DM architectures, called PV-DBOW and PV-DM, incorporate paragraph's identifier. This allows the architectures to store the information about the input paragrahs, which is then used as the paragraph's embedding. The final paragrpah embedding is linear combination of the representations of both architectures. Note that, a paragrpah can be any piece of continuous text.

TODO: my own graphic here

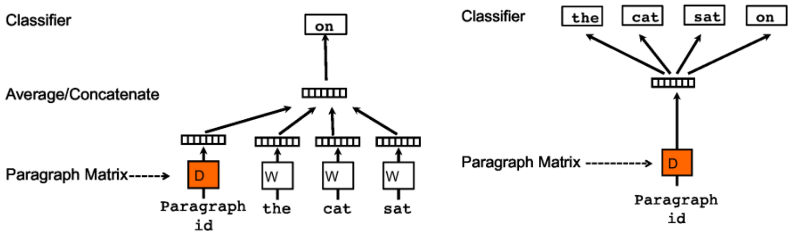


Figure 2.1: PV-DM and PV-DBOW architectures.

Paragraph Vector is trained using language modelling — both architectures should predict a word which is probable in the given context. In PV-DM the context is paragraph id and the neighbouring words, in PV-DBOW the context is only the pragraph id.

Paragraph Vector's advantage is its small size and therefore quick learning. Moreover it is able to process paragraphs of all lengths. The disadvantage is that the embedding must be learned even during inference.

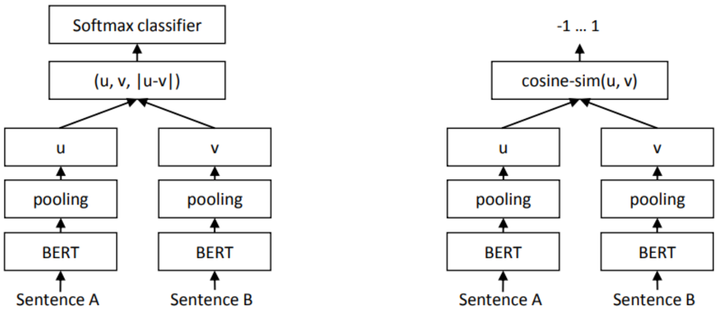


Figure 1: SBERT architecture with classification objective function, e.g., for fine-tuning on SNLI dataset. The two BERT networks have tied weights (siamese network structure).

Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

Figure 2.2: SBERT architecture with siamiese networks.

2.1.2 SBERT

Sentence-BERT (or SBERT for short) introduced in Reimers and Gurevych [2019], is a composition of a BERT-like model with pooling layer above its final hidden states. This architecture is common for sequence classification using a transformer model. SBERT differs from these simpler approaches, by finetuning on NLI datasets using siamiese networks. The training setup is depicted in Figure 2.2. After such training the STS scores of SBERT embeddings significantly increases.

The disadvantage of SBERT is its inability to process longer pieces of texts. The workaround is to either truncate the input or to average multiple embeddings produced by a sliding window over the input. Both approaches limit the model's ability to see the document as a whole, which could impair the quality of the produced embeddings.

2.1.3 Longformer

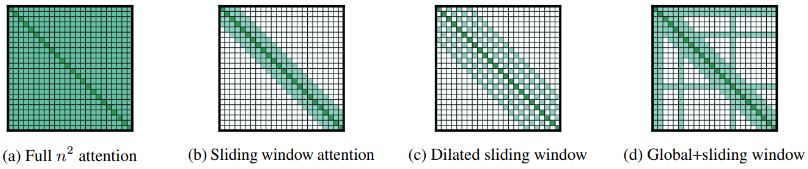
Longformer introduced in Beltagy et al. [2020], is a transformer with sparse attention matrix. Whereas in traditional transformer we see dense attention matrix — every token “attends” to every other token, in Longformer's attention every token “attends” only to selected few global tokens and neighbouring tokens. Example of such sparse attention matrix is depicted in Figure 2.3. This allows Longformer to process inputs in linear time of the input length. Thus Longformer is able to process inputs up to 4096 tokens long.

TODO: maybe comparison to normal dense attention

While sparse attention matrix allows Longformer to process longer texts, it also limits its computational power. Zaheer et al. [2020] show that with sparse attention we need more layers to match the power of a dense attention matrix.

To produce input embeddings we average the embeddings of last hidden states.

TODO: training



(a) Full n^2 attention (b) Sliding window attention (c) Dilated sliding window (d) Global+sliding window

Figure 2: Comparing the full self-attention pattern and the configuration of attention patterns in our Longformer.

Figure 2.3: Longformer attention matrix.

2.1.4 BigBird

TODO: how is it different from longformer

3. My model

- what models I have tried
- training ideas

4. Evaluation

In this chapter we will describe a set of benchmarks, which will test our model and enable us to compare it to other models. First we will describe the tasks — datasets and corresponding evaluation metrics, then we will talk about the models. Results of the benchmarks are discussed in Chapter 5.

4.1 Tasks

Each task aims to test a different aspect of a model. Our aim was to design a set of tasks, which can capture a model's capability to embed whole documents. The major obstacle we faced was the lack of labeled datasets with longer pieces of text (more than 512 tokens).

TODO: how did we solve the issue

TODO: complete list of task types

Classification

Classification tasks test model's capability to separate inputs based on a complex feature. In our settings, classification tasks can tell us what information the document embedding contains.

4.1.1 IMDB Sentiment Analysis

IMDB sentiment analysis task is a simple binary classification task. The dataset contains movie reviews from the Internet Movie Database[1] labaled as either positive or negative. The dataset is commonly referred to as IMDB classification or sentiment dataset Maas et al. [2011].

The dataset is split evenly to test and train set, each having 25000 reviews. The dataset also contains 50000 unlabeled reviews. The label distribution in both sets is uniform, each of of the two labels is represented by 12500 reviews.

As can be seen from the figure Figure 4.1 the reviews are quite short with only 13.56% being longer than 512 RoBERTa tokens.

We included this task to see how our model compares in realtively undemanding settings, while also evaluating its performence on shorter documents.

[1] www.imdb.com

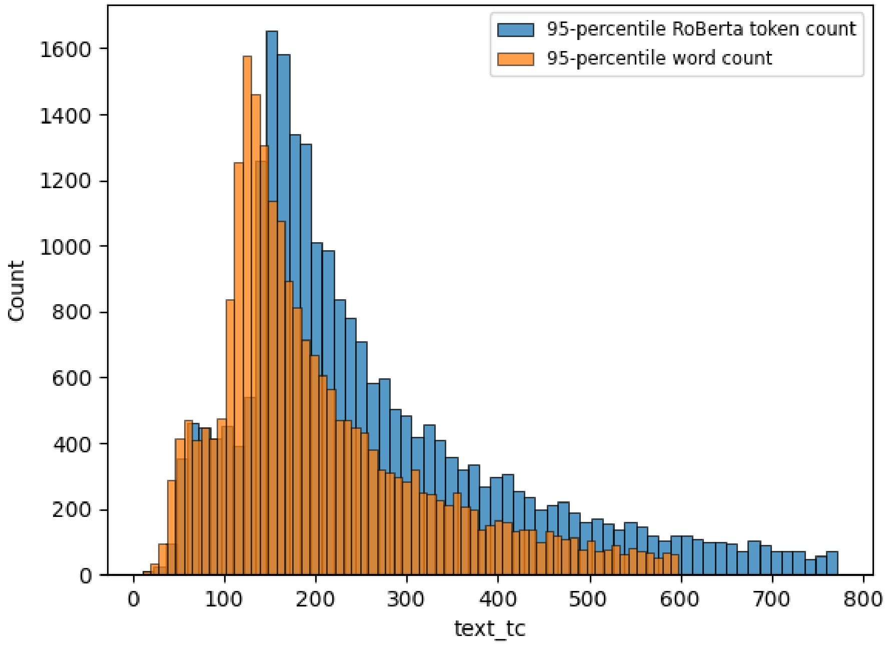


Figure 4.1: Word count and token count distribution of 95-percentiles of reviews. The tokens are generated using RoBERTa's pretrained tokenizer from HuggingFace

5. Results

- tabulated experiment results
- discussion

Conclusion

- what i have done — what are the model's results
- other findings

Bibliography

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `https://aclanthology.org/P11-1015`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

List of Figures

List of Tables

A. Attachments