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 Intelligence

Prague 2023

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 **2**

1 Document representation **3**
1.1 Desirable qualities of document representations . . . 3
1.1.1 Text understanding . . . 3
1.1.2 Maximum input length . . . 4
1.1.3 Combining depth and breadth . . . 4

2 Background **6**
2.1 Text embedding . . . 6
2.2 Paragraph Vector . . . 6
2.3 SBERT . . . 7
2.4 Sparse attention . . . 7
2.4.1 Longformer . . . 7
2.5 Teacher-student training . . . 8
2.6 Canonical Correlation Analysis . . . 8
2.6.1 CCA as a loss . . . 9
2.6.2 Deep CCA . . . 9
2.6.3 Soft CCA . . . 9
2.7 Contrastive loss . . . 10

3 Related Work **11**

4 My model **13**

5 Evaluation **14**
5.1 Tasks . . . 14
5.1.1 IMDB Sentiment Analysis . . . 14

6 Results **16**

Conclusion **17**

Bibliography **18**

List of Figures **20**

List of Tables **21**

A Attachments **22**

Introduction

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

1. Document representation

We dedicate this chapter to the center point of our thesis – representing documents with vectors. The goal of this chapter is to define the desired qualities of our embeddings and illustrate them on examples.

1.1 Desirable qualities of document representations

Since there are many ways how to embed a document we deemed it necessary to describe what are the target qualities of the embeddings. There are two dimensions along which we asses document representations: *depth* and *breadth*. *Depth* dimension defines how deep is the model's understanding of the input text. *Breadth* dimension defines the amount of information the model is able to process. These dimensions are in a sense contradictory to each other. We cannot expect a fully learned model of a fixed size to deepen its understanding of the input while maintaining the same maximum input size. Thus we are searching for a ideal ratio of these two qualities.

1.1.1 Text understanding

In the depth dimension we study how well the model understands its input. For an input *"This is a really nimble horse. You should be cautious when riding him."* we would like the model to understand that *"him"* refers to the *"nimble horse"*. Or that the meaning of *"stick"* in the following sentences is largely different:

- *"You are really good at tennis. You should stick to it."*
- *"Some dogs really like to fetch a stick, some just carry it."*

To produce embedding that would reliably reflect the input's meaning, we expect the model to see the words in context of their relationship to other words. To do so, architectures that enable deep text understanding allow the model to compute with relationships of ordered subwords, words or sequences of words. Such architectures are able to put each word in relevant context and thus recover all the meanings of each sequence of words if they are given enough training data.

To compute with relationships of ordered subwords, the neural network must connect representations of two subwords such that the connections also carry the information about the subwords' ordering. The same holds for computing with relationships of ordered words or sequences of words.

To give some examples we can pick some representative embedding models along the depth dimension going from the ones that show the shallowest understanding of their input to those who show the deepest. At the beginning, there are models that ignore both word ordering and their relationships. An example of such model is TF-IDF. If we go deeper we run into Paragraph Vector Le and Mikolov [2014], that takes word relationships into an account, but ignores word ordering to a certain degree. Next there are models with architectures based on one-directional or two-directional RNNs. These models compute with ordering of words implicitly, but do not have the capacity to consider some word relationships directly. Finally there are models that are based on the architecture of Transformer Vaswani et al. [2017]. Such models not only take the word ordering into consideration, but are also able to compute over all relationships between input's text units. An example of such architecture is SBERT Reimers and Gurevych [2019], which is essentially an instance of BERT Devlin et al. [2019] finetuned on NLI datasets to encourage deeper text understanding.

1.1.2 Maximum input length

In the breadth dimension we study the maximum context length of the model. Put simply, we study the maximum number of tokens the model is able to process. The longer the maximum context length is, the longer text we can coherently embed.

Model's context length is especially important when we expect the properties of the piece of text to vary throughout it. For such texts the embedding of its beginning would be different than the embedding of its end or of the whole text. Comparing embeddings of long texts, thus requires to consider the text as a whole and embed it with a model whose maximum context is larger or equal than the given text's length.

If we revisit the models mentioned in the previous section, we realise the order of the mentioned models flips. The models with the smallest maximum context are Transformers with classical attention mechanisms. They are typically not limited by their architecture, but by the memory they consume for longer inputs. The large memory consumption originates in the classical attention mechanism, which takes into account all of the n^2 relationships among the n input tokens. Though recently there have been many improvements to the implementation of classical attention(TODO: ref) which improves their efficiency, and there are other attention mechanisms which offer more efficient alternative, Transformers are nevertheless costly for longer inputs both in terms of memory and time. Next type of models are those built using RNNs. Though RNNs can in theory process unlimited number of tokens, in practice they are known for "forgetting" past tokens, once the input is long enough (TODO: ref). On the other hand, Paragraph Vector or TF-IDF can both process in theory limitless number of words. In practice Paragraph Vector is limited by the dimension of the document embedding, which can be, however, scaled arbitrarily.

1.1.3 Combining depth and breadth

As we have hinted above, the goal of this thesis is to combine deep understanding with longer maximum contexts. Combination of both qualities would result in a model that can not only register and compute with relationships of words, but do so over large gaps of unrelated tokens. For example, in *"I really like your T-shirt. It almost looks as if it was designed by some really alternative guy living in some cave in Gran Canaria. You should definitely wear it more often."*, we would expect such model to connect *"it"* in the last sentence with the *"T-shirt"* from the first sentence.

To find a compromise, we combine models of the two extremes: models like Transformer that can understand the input very well and models like Paragraph Vector which can process very long sequences.

2. Background

In this chapter we describe model architectures, concepts and training methodologies our work uses.

2.1 Text embedding

To allow neural networks to process text, each continuous piece of text is mapped to a series of vectors that represent it. This mapping and its result is called an embedding. Traditionally we would split the input text to sub-words and allow the network to change their embedding arbitrarily to solve some end task. Example of such task is sub-word prediction, where the network guesses masked-out sub-word.

In our work we train a network to map a long piece of text to a single vector. As the size of the embedding is limitted, we expect the network to effectively compress the input. In this work our aim is not to reconstruct the input from the embedding, but rather to capture both the structure and meaning of the text.

2.2 Paragraph Vector

Paragraph Vector Le and Mikolov [2014] (also known as Doc2Vec) is a text-embedding model that is an extension of Word2Vec Mikolov et al. [2013]. Paragraph Vector is composed of two sub-models called Distributed Memory (or DM) and Distributed Bag of Words (or DBOW). Both DM and DBOW are trained on token prediction. To predict masked-out tokens, both models use embedding of the whole input, while DM additionally computes word embeddings. In practice, each embedded piece of text (a word or an input) gets an identifier, which is used as an index into an embedding matrix that stores the vector representations.

Paragraph Vector, thanks to its simplicity, can be trained quickly, even with limited computational resources. Additionally the model can embed texts of any length. However, Paragraph Vector has three major disadvantages. First, it ignores text structure, by assuming the input is a bag of words rather than its sequence. Second, because it is very simple model, it does not have the capability to understand complex relationships of words, sentences or paragraphs. Third, due to how embeddings are computed, the inference time is very long, since it is necessary to train the model until convergence on the previously unseen test input.

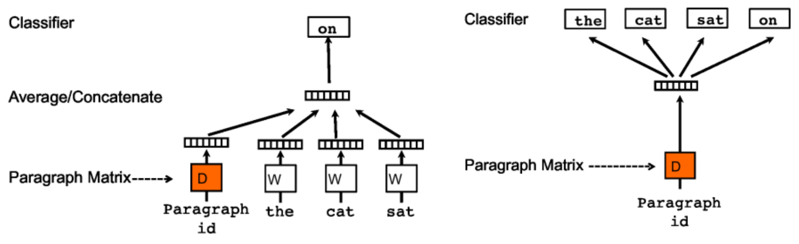

Figure 2.1: PV-DM and PV-DBOW architectures.

2.3 SBERT

Sentence-BERT Reimers and Gurevych [2019] (or SBERT) a composition of a Transformer (TODO: citation) encoder with a pooling layer above its final hidden states. The model is warm-started from a pre-trained Transformer such as BERT (TODO: citation), RoBERTa (TODO: citation) or MPNet (TODO: citation). Then it is fine-tuned on NLI datasets so that the pooled representations of the whole inputs are comparable using cosine. The goal is that a semantic difference between two inputs should correspond to the cosine between the two pooled representations.

Thanks to higher parameter count, SBERT can understand more complex relationships than Paragraph Vector. As no parameter adjustment is necessary during inference, SBERT predicts embeddings very quickly which makes it suitable for search applications. The disadvantage is that it can process text only up to a certain length and unfortunately the same training cannot be used for longer texts due to lack of appropriate datasets.

2.4 Sparse attention

Traditional Transformer encoder uses full-attention, which means that every token can interact (or attend) to any other token. This type of attention creates quadratic number of interactions to input length and therefore also quadratic time and space complexity. This seriously limits use of transformers for longer inputs even with efficient cuda kernels (TODO: really? Citation needed). To overcome this issue, interactions between some tokens are ignored. Such type of attention is called sparse.

TODO: My graphic of attentions

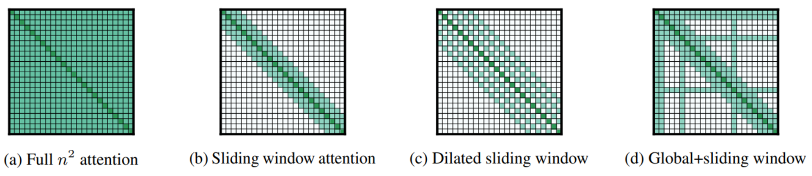
(a) Full n^2 attention
(b) Sliding window attention
(c) Dilated sliding window
(d) Global+sliding window

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

Figure 2.2: Longformer attention matrix.

Sparse attention can have different designs but they are usually composed of several types of mechanisms (such as sliding window, global, random – I can write more about this).

2.4.1 Longformer

Longformer Beltagy et al. [2020], is a transformer with sparse attention that combines global and sliding window mechanisms. It is warm-started from RoBERTa weights with its learned embeddings copied 4 times. Thus Longformer can process inputs up to 4096 tokens long. Then the model was fine-tuned using Masked

Language Modelling (or MLM) on a dataset of long documents such as Wikipedia articles or books.

There are many others sparse attention models, but we ephasize Longformer since it is one of the best performing Tay et al. [2020] and yet has relatively simple attention mechanism, which makes it a suitable benchmark.

2.5 Teacher-student training

Teacher-student training is a setup in which we train a model (a student) according to the relationship of its output with outputs of another model (a teacher), which is frozen. The desired relationship can differ, but often we aim to minimize the difference between the two outputs to simulate the teacher model by the student. Teacher-student training can been used to dramatically decrease the size of a model, while at the same time only slightly decreasing its performance. Another use of this method is in multi-modal learning when more modalities of the same input are available. In such cases we can have several teacher models, each for different modality, providing their outputs to the loss of a single student model.

TODO: some graphic

2.6 Canonical Correlation Analysis

Canonical Correlation Analysis (CCA) computes linear projections of two vectors such that their projections are maximally correlated. To define CCA properly, let us look at its mathematical formulation.

Definition 1 (Canonical Correlation Analysis)**.** *For two matrices $X_1 \in \mathbb{R}^{n_1 \times m_1}$ and $X_2 \in \mathbb{R}^{n_2 \times m_1}$, Canonical Correlation Analysis finds $p \in \mathbb{R}_1^m$ and $q \in \mathbb{R}_2^m$ that maximize*

$$\begin{aligned} corr(X_1 p, X_2 q) &= \frac{p^T X_1^T X_2 q}{||Xp|| ||Yq||} \\ s.t. \quad ||X_1 p|| &= ||X_2 q|| = 1 \end{aligned} \tag{2.1}$$

Definition 1 suggests that CCA gives only a single number as the measure of correlation of two matrices. When the dimensions of the input vectors are large enough, however, often there is a some combination of features that results in correlation of 1. This would make CCA relatively useless in context of high-dimensional spaces. To overcome this issue we assume multiple correlations for several mutually orthogonal projections. Again, let us look at the mathematical formulation.

Definition 2 (Canonical Correlation Analysis in more dimensions)**.** *For two matrices $X_1 \in \mathbb{R}^{n_1 \times m_1}$ and $X_2 \in \mathbb{R}^{n_2 \times m_1}$, Canonical Correlation Analysis in k dimensions finds $P \in \mathbb{R}^{m_1 \times k}$ and $Q \in \mathbb{R}^{m_2 \times k}$ that maximize*

$$\begin{aligned} &\sum_{i=1}^{k} corr(X_1 P_{*i}, X_2 Q_{*i}) \\ s.t.\quad & P^T X_1^T X_1 P = I_k = Q^T X_2^T X_2 Q \end{aligned} \tag{2.2}$$

If we consider the conditions in Definition 2, the resulting value can be easily formulated as $trace(P^T X_1^T X_2 Q) = trace(P^T \Sigma_{X_1 X_2} Q)$, where $\Sigma_{X_1 X_2}$ is covariance matrix of X_1 and X_2.

TODO: analytical solution to pure CCA

2.6.1 CCA as a loss

CCA can be used as a loss in teacher-student training to establish some sort of correspondence between the teacher's and the student's outputs. When CCA is maximized the teacher's and the student's outputs are first linearly transformed, before the correlation is computed. CCA is therefore weaker requirement than classical correlation.

The problem of using CCA as a loss is that it is defined in the context of the whole dataset rather than just a minibatch. It is therefore unclear how CCA for a pair of minibatches of vectors should be computed. Someone (TODO: citation) found that using large enough batch size is sufficient for the training to converge.

2.6.2 Deep CCA

Deep CCA (or DCCA) is an extension of CCA that computes projections using neural networks. As such it is more powerful than plain CCA as it allows for non-linear projections. To compute DCCA the network has to be trained on the pairs of vectors with CCA as its loss.

TODO: graphic of architecture

If CCA is weaker condition than just correlation, DCCA is even weaker since there is no limit to how the projections should look like.

2.6.3 Soft CCA

Soft CCA reformulates CCA and thus allows its straightforward use in the context of minibatches. With constraints from Definition 2 taken into account, CCA can be formulated using Forbenious matrix norm:

$$P^*, Q^* = \underset{P,Q}{\operatorname{argmin}} ||X_1 P - X_2 Q||_F^2 \tag{2.3}$$

$$= \underset{P,Q}{\operatorname{argmin}}\, trace\Big((X_1 P - X_2 Q)^T (X_1 P - X_2 Q)\Big) \tag{2.4}$$

$$= \underset{P,Q}{\operatorname{argmin}} -2 trace(P^T X_1^T X_2 Q) \tag{2.5}$$

$$= \underset{P,Q}{\operatorname{argmax}}\, trace(P^T X_1^T X_2 Q) \tag{2.6}$$

$$= \underset{P,Q}{\operatorname{argmax}} \sum_{i=1}^{k} corr(X_1 P_{*i}, X_2 Q_{*i}) \tag{2.7}$$

So, in essence minimizing CCA is the same as minimizing the difference between projections, whose features are decorrelated. This is the formulation Soft CCA builds on. Soft CCA decomposes CCA into to parts:

- minimization of the difference between projections

$$||X_1P - X_2Q||_F^2 \tag{2.8}$$

- decorrelation of each projection P

$$\sum_{i \neq j}(P^T X_{mini}^T X_{mini} P)_{ij} = \sum_{i \neq j} \Sigma_{X_{mini}P}, \tag{2.9}$$

  where X_{mini} is batch-normalized minibatch.

To bring correlation matrix of the current minibatch $\Sigma_{X_{mini}P}$ closer to the true covariance, the decorrelation part is in fact computed from a covariance matrix that is increamentally learned.

In this way Soft CCA increamentally decreases CCA through incrementally learned approximation of the projections' covariances.

2.7 Contrastive loss

Contrastive losses are types of losses which compare several model's outputs to each other and based on this comparison they compute a value, which can be used as a loss. Contrastive loss is often used when the model's goal is to understand and distinguish different inputs. In this setting, we would penalize the model if the representations of similar inputs are far apart or if the representations of different inputs are close. Similar inputs are called positives, wheras distinct inputs are called negatives.

When no supervised information about the similarity of a pair of inputs is available, we can use in-batch outputs as negatives. Here we automatically assume that all our inputs are distinct and that the model should distinguish between all of them.

3. Related Work

Describe very broadly other approaches to embedding documents. What has been achieved rather than how some methods work.

Current shortcomings:

1. minimal resources about teacher-student (de-facto only SBERT)
2. no resources on multi-modality
3. few papers dedicated strictly to embedding documents
4. fill-in my resources about datasets

- Efficient transformers
  - Attention mechanisms
    - Efficient Transformers Tay et al. [2022] – nice overview of efficient transformers
    - Longformer Beltagy et al. [2020] – mention it
    - BigBird Zaheer et al. [2020] – probably has some insights both about sparse attentions (only cover very briefly) and about document processing
  - Attention implementation
    - Flash Attention 2 – mention it, just to show there are approaches that completely ditch sparse attentions (such as LLama2 Long)
    - xFormers – dtto
    - Llama 2 Long – bringing long context to the extreme, only uses full attention
  - Completely changing the architecture
    - Siamese Multi-depth Transformer-base Hierarchical (SMITH) encoder Yang et al. [2020] – approach to long inputs using hierarchies rather than changing type of attention
    - Hierarchical Attention Network (HAN) Yang et al. [2016] – hierarchical approach along text objects (words, sentences, paragraphs) for document classification
- Training approaches – should cover contrastive loss, teacher-student and other types of training
  - Paragraph Vector Le and Mikolov [2014] – just mentioned it. It is already surpassed methodology.
  - SBERT Reimers and Gurevych [2019] – did pretty much the same as us (took a model and finetuned it so that the produced embeddings are good), except for sentences

  - MPNet Song et al. [2020] – best SBERT model and the model we are working with
  - That paper where SBERT is finetuned on low-resource language with teacher-student training
  - OpenAI embedding paper – use contrastive loss, mention large batches and the memory constraints it brings
  - Specter Cohan et al. [2020] – focused on scientific documents, use of contrastive loss based on citations
  - Transformer based Multilingual document embedding model Li and Mak [2020] – transformer version of LASER, embedds documents
  - LASER Schwenk and Douze [2017] – the idea that representations in different languages should be close to each other, probably not so relevant because they are sentences and also my work only deals with english

- Unorthodox document embedding approaches

  - Self-Supervised Document Similarity Ranking Ginzburg et al. [2021] – really focused on generating semantically meaningful representations of documents, but with an extra cost and complexity
  - Cross-Document Language Modelling Caciularu et al. [2021] – uses Longformer for cross-document task. Illustrates that Longformer can be flexible and useful for document processing.

- Evaluation datasets

  - Wikipedia Similarities
  - ...

4. My model

- what models I have tried
- training ideas

5. 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 6.

5.1 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.

5.1.1 IMDB Sentiment Analysis

IMDB sentiment analysis task is a simple binary classification task. The dataset contains movie reviews from the Internet Movie Database[1] labeled 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 5.1 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 relatively undemanding settings, while also evaluating its performance on shorter documents.

[1] `www.imdb.com`

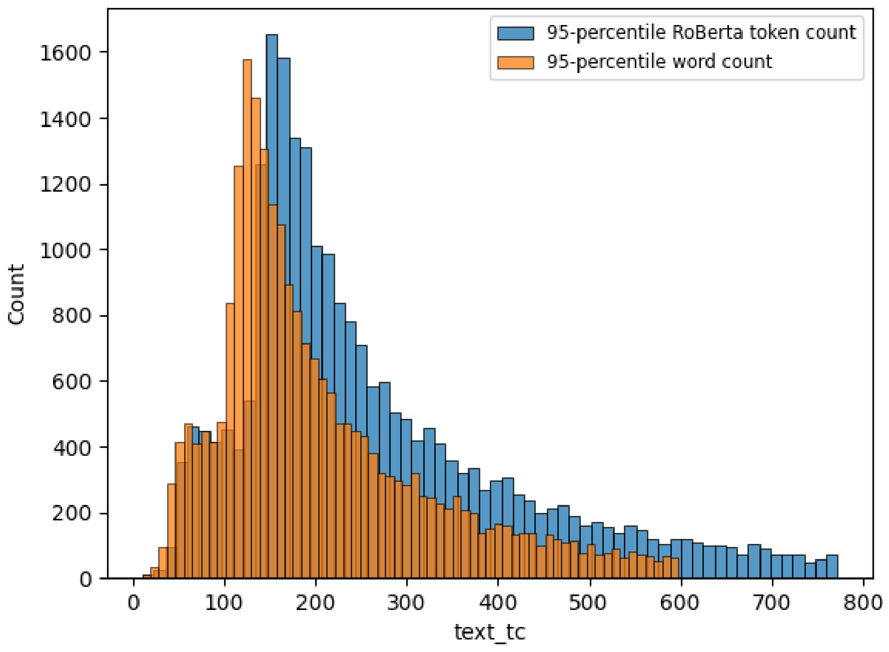

Figure 5.1: Word count and token count distribution of 95-percentiles of reviews. The tokens are generated using RoBERTa's pretrained tokenizer from Hugging-Face

6. 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.

Avi Caciularu, Arman Cohan, Iz Beltagy, Matthew E Peters, Arie Cattan, and Ido Dagan. Cdlm: Cross-document language modeling. *arXiv preprint arXiv:2101.00406*, 2021.

Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Dvir Ginzburg, Itzik Malkiel, Oren Barkan, Avi Caciularu, and Noam Koenigstein. Self-supervised document similarity ranking via contextualized language models and hierarchical inference. *arXiv preprint arXiv:2106.01186*, 2021.

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

Wei Li and Brian Mak. Transformer based multilingual document embedding model. *arXiv preprint arXiv:2008.08567*, 2020.

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.

Holger Schwenk and Matthijs Douze. Learning joint multilingual sentence representations with neural machine translation. *arXiv preprint arXiv:1704.04154*, 2017.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL `https://doi.org/10.1145/3530811`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1725–1734, 2020.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

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

2.1 PV-DM and PV-DBOW architectures. 6

2.2 Longformer attention matrix. 7

5.1 Word count and token count distribution of 95-percentiles of reviews. The tokens are generated using RoBERTa's pretrained tokenizer from HuggingFace 15

List of Tables

A. Attachments