# Narcissist! Do you need so much attention?

Gaëtan Caillaut, Nicolas Dugué, et Nathalie Camelin

Université du Mans, LIUM

## Abstract

After the rise of `Word2vec` came the `BERT` era, with large architectures allowing to deal with polysemy by taking into account the contextual information, leading to great performance improvement on classic NLP tasks. `BERT` systems are considered universal: they can be fine-tuned to address any task efficiently. However, these systems are huge to deploy, not trivial to fine-tune, and may not be fitted to some corpora, *e.g.* domain-specific and small ones. For instance, we consider the DEFT 2018 corpus of tweets and show that `CamemBERT` is not appropriate to this corpus and task. According to the Occam's razor principle, we thus designed `MiniBERT`, a tiny `BERT` architecture that includes a simplified self-attention mechanism and does require neither pre-training, nor external data. We show that this easily trainable and deployable system obtains encouraging results on DEFT, whilst providing interpretable results.

**Mots-clef**: `BERT`, `MiniBERT`, contextual embeddings, explainable AI, DEFT.

## 1 Introduction

`BERT`-like architectures allowed a great rise of the performances of NLP systems in many classic tasks. However, in this paper, we aim to question the systematic use of these architectures in NLP. Indeed, even if there is a claim that these architectures are universal, and fine-tuning them is, most of the time, a good solution, we argue that *ad hoc* systems specialized on the task and corpora we consider may sometimes be more fitted. Indeed, `BERT` systems are huge, fine-tuning and deploying them is not trivial. Furthermore, the hugeness of these systems makes them uninterpretable. Also, generic systems may not be fitted to small domain-specific corpora on which internal representations of the pre-trained architectures are not relevant.

After having narrated the tale of contextual word embedding in Section 2, we describe the `BERT` architecture in Section 2.3 and in particular, the self-attention mechanism. The usage of such huge architecture is questioned in Section 3, in particular in the case of small domain-specific corpora, and we detail accordingly `MiniBERT`, our Occam's razor system: a refined architecture with only one head of a simplified self-attention mechanism we introduce, low-dimensional embeddings, fixed positional embeddings. By using data of the DEFT 2018 contest [PGB+18], we show in Section 5 that this refined architecture is relevant in the case of this small and specific corpus. It obtains encouraging results while being easy to train: it indeed requires neither pre-training, nor external data.

## 2 From `Word2vec` to `BERT`

### 2.1 Word embeddings.

We choose to start this tale of word embeddings with `LSA` (Latent Semantic Analysis), which is, as far as we know, the first famous model to use an abstract vector space to represent *meaning* (in this case, mostly of documents). This model is based on the matrix factorization of the document-term matrix representing a corpus, this factorization being based on `SVD` (Singular Value Decomposition). Then, with generalized `LSA`, Matveeva et al. [MLFR05] introduced a similar model, which allows to obtain word embeddings. The authors proposed to apply `PMI` (Pointwise Mutual Information) on a term-term co-occurrence matrix, extracted using a sliding window, and to extract 200/300 dimensional word vectors using `SVD`. Such work is closely related to the Levy and Goldberg's one ten years later [LGD15]. And, as one can see, the parameters that are commonly used today, such as the size of the sliding window or the size of the vectors, were the same in 2005.

In 2008, Mnih and Hinton [MH09] described a pioneer neural language architecture, a log-bilinear model learning the probability of a word to appear in a context. It can be seen as a precursor of the `Word2vec`

model detailed by Mikolov et al. [MCCD13], that is also trained with a hierarchical approach. Finally, in 2014, Levy and Goldberg [LG14] related neural models to matrix factorization, bridging the gap between neural models and `SVD` based approaches, and thus coming full circle.

However, with these approaches, there is one and only one vector to represent a word, while a word may have distinct senses, *e.g.* bank can be a financial institution or a piece of land aside of the river.

## 2.2 Multi-prototype word embedding.

With Gaussian embeddings [VM14], a word is not only represented by a vector, but also by its variance on each dimension, thus mapping a word to an area in the resulting space. This framework allows to take into account the variety of contexts a word appears in, and it is thus the first attemp to take into account polysemy. However, this approach is not capable of mapping a vector to each distinct sense of a word. Multi-prototype approaches, that are then discussed, allow to map each sense of a word to a distinct vector. The first approach we discuss is the one proposed by Huang et al. [HSMN12], which is based on a two-pass algorithm. The first step consists in learning classic single-prototype word embeddings. Then, in a second step, authors introduced context vectors as follows: the context vector of a word occurrence is the mean of the vectors of its context. Then, all the context vectors of the occurrences of a same word are provided as inputs of a clustering algorithm to group occurrences sharing similar contexts, and thus, according to the distributional hypothesis, the same meaning. This clustering thus serves the purpose of a `WSD` (Word Sense Disambiguation) algorithm. Then, the corpus is annotated so that occurrences of same (resp. distinct) senses can be processed together (resp. separately). Finally, a word embedding learning algorithm is executed, and because the corpus is now annotated with the different senses of each word, the algorithm maps a distinct vector to each sense. In a same spirit, the Tian et al. [TDB+14] paper introduced an `EM` (Expectation-Maximization) algorithm that maps word occurrences to senses, and learns embeddings for each sense. The algorithm iterates through two steps until convergence: learning word embeddings assuming a fixed occurrence to sense mapping, and mapping occurrences to senses assuming vectors for each sense. Li and Jurafski [LJ15] introduced a one-pass algorithm based on the *Chinese restaurant process* in the same spirit. However, all of these multi-prototypes approaches suffer from the same flaws as clustering: the research space is huge, the number of clusters is a critical parameter, and there are a lot of local extrema. Thus, contextual embeddings were then introduced to deal with the meaning of the words according to their context, and thus deal, among other things, with homonymy and polysemy.

## 2.3 Contextual word embedding.

In this framework, words are no longer associated to a discrete, finite, number of senses. Instead, a word's meaning, its embedding, is tied to its context. Since there is possibly an infinite number of occurrences of a same word, there is also a possibly infinite number of vectors representing this same word, but in different contexts. In the remaining, we consider Transformers [VSP+17] based contextual embeddings, that were proven to be efficient in a wide variety of downstream NLP tasks. The main concept of the transformer-based contextual embeddings framework, such as `BERT` [DCLT18], is related to the multi-prototype learning process described by Huang et al. They calculate, for each word, its context vector as the mean of the vectors representing the words in its context. Contextual embeddings are calculated according to a function of the embeddings of their context too, but this function is most-of-the-time NOT the mean function. In transformers, this function is actually learnt using supervision (*masked language model*), and it is denominated as the *self-attention mechanism*.

**Masked language model.** The Masked Language Model (MLM) is really similar to the *Continuous bag of words model* (`Cbow`) of `Word2vec`. The idea is to maximize the following likelihood for each token $w_i$ in the ordered set of the $n$ corpus tokens: $\prod_{i=1}^{n} p(w_i \mid w_{i-sl}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+sl})$, where $sl$ is the sequence length, a model parameter. Basically, we aim to predict the *target* word (or *key* word) $w_i$ considering its *context* (the *query*). However, in the transformers implementation, the model strongly relies on the self-attention layer, the core of these architectures.

**Self-attention mechanism.** As far as we know, self-attention was first introduced in [YYD+16] in a simple way. The self-attention mechanism of [DCLT18] takes an embedding matrix as input. Usually, positional information is injected by summing the input word-embeddings matrix and a position-embeddings matrix, which can be either trained or fixed (according to an *ad hoc* mathematical function). Then, the core of the self-attention mechanism is made of three

matrices (Fig 1), corresponding to three different projection functions, $\mathcal{Q}$ (Query), $\mathcal{K}$ (Key) and $\mathcal{V}$ (Value). The first matrix is applied to the query, *i.e.* to the embedding vectors of the words of the context $(w_{i-sl}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+sl})$. The second one is applied to the key word $w_i$ to predict. This allows to obtain a vector of similarity between $w_i$, as the key, and all the query, or context, words. This is somehow similar to the $U$ embedding matrix and the $V$ context matrix of `Skip-gram`. Then, this vector of similarity is normalized by applying the *softmax* function, producing an attention vector, which are basically weights indicating how much the key word attends to each word of its context. The contextual vector of this key vector is then calculated as the matrix product between the attention vector and the embedding matrix projected by $\mathcal{V}$, which is similar to computing the average value embeddings weighted by the attention. We argue that the way Huang et al. calculate the context vector of a word is similar to an attention mechanism attending uniformly to each word of the context.

# 3 Too much attention

## 3.1 Questioning `BERT` architectures

The most common architectures, such as `BERT-Base`, `RoBERTa-Base` or `CamemBERT`, encompass 12 attention layers, each layer being made of 12 distinct attention heads, with embeddings of size 768. Training 144 attention heads surely requires a lot of data and GPU time, thus a lot of power. It is a concern at the dawn of a possible energy transition era. Moreover, resources required to train today's hugest models, for example `GPT-3`, are not within range of everyone. However, as shown by Clark et al. [CKLM19], training many attention-heads allows each head to specialize itself by attending to specific parts of the sentence. For instance, some attention heads are trained such that direct objects attend to their verbs, or prepositions attend to their objects. Most of these mechanisms are very useful to capture both syntactic and semantic aspects of a sentence. Such huge pre-trained models are thus very powerful, since they allow to obtain state-of-the-art results on a wide range of NLP tasks by relying on one single model. Indeed, such pre-trained systems can then be fine-tuned to be used on specific tasks on corpora. However, one may wonder whether it is necessary to fine-tune that many (144!) self-attention heads: while it is a simpler task, it still requires a lot of GPU time. Furthermore, fine-tuning such a huge system is not straightforward: for instance, how many epochs are actually required to specialize the system to the new data? Also, such complex systems with so many attention layers may not be relevant to deal with specific tasks. For instance, an attention head that makes passive auxiliary attending to the verb they modify may not be necessary to disambiguate a word's meaning, or to do sentiment analysis. Actually, Clark et al. [CKLM19] showed that many attention heads are doing nothing on purpose, because they are not required at all. Other work have shown that the ablation of half of `BERT`'s attention heads results in a very slight decrease in performances [GDA20], suggesting that such models are oversized. De Wynter and Perry [dWP20] even showed that significantly smaller architectures could beat `BERT-Large`, indicating that training smaller models could be beneficial at all levels. Furthermore, we argue that the large corpora used to pre-train these systems may not be adapted to train models to deal with highly specific topics (different vocabulary, or same vocabulary but different meanings), where only few data is made available to fine-tune the system. For example, a model trained on Wikipedia may be irrelevant when applied on a medical dataset. Finally, these huge systems lack interpretability, partly because of their tremendous size [Rud19]. Meanwhile, being able to understand, if necessary, how and why the model produces its outputs seems crucial in critical fields where AI is involved, such as medical or legal domains. Some work [CKLM19] have been carried out to try to understand what `BERT` does, but it consists in probing individual attention heads which only give an insight on local behaviors of the model. For instance, it is hard to tell how errors, or undesirable human biases are propagated through the whole network.

These concerns raise various questions. One may ask if the number of attention layers could be reduced while preserving performance. Also, it is not clear whether training three projection matrices ($\mathcal{Q}$, $\mathcal{K}$and $\mathcal{V}$) is truly required. We investigate further these questions considering transformer based models in the next section.

## 3.2 `MiniBERT`'s architecture

`MiniBERT` is an attempt at refining the `BERT` architecture. We experimented by removing, from `BERT`, everything that may not be necessary in our context in order to keep only what is really useful.

We first removed dropout layers, as well as residual connections. In our experiments on dataset described further, these layers seem unnecessary. But our work mostly focuses on the encoding layers: from 12 self-attention layers, each made of 12 attention heads,

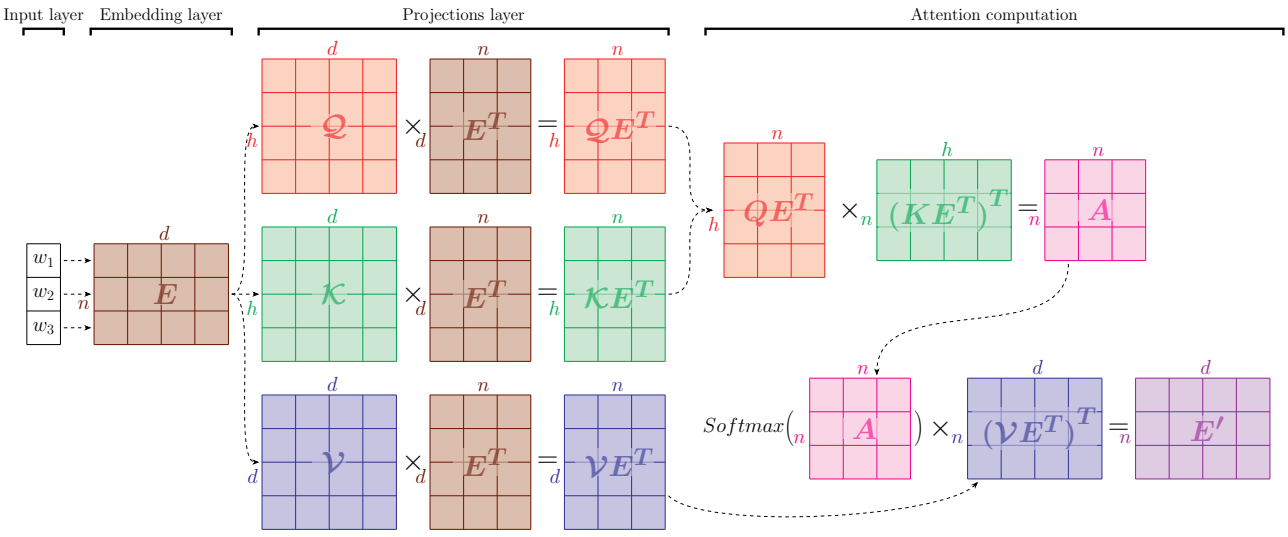


Figure 1: Computation of the attention for an input sequence of size $n = 3$, embedding dimension $d = 4$ and hidden dimension $h = 4$. $A$ is the attention weights matrix and $E'$ are the resulting contextualized embeddings.

we aim to converge to the lightest system. Our first `MiniBERT` model hence consists of one embedding matrix, the three projection matrices $\mathcal{Q}$, $\mathcal{K}$ and $\mathcal{V}$, and some prediction layers, as in `BERT`. We wisely called this architecture `MiniBERT` **self-attention** (`MiniSA`), since it is a minimal version of `BERT` with only one self-attention head. An overview of this architecture is given in Figure 1.

The `MiniBERT`'s language model aims to maximize the likelihood of a word given its context: $\prod_{i=1}^{n} p(w_i \mid w_{i-sl}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+sl})$, where $n$ is the number of tokens in the corpus, and $sl$ is the sequence length. It can be written as follows if we consider the **transforming** (with $\mathcal{Q}$, $\mathcal{K}$, $\mathcal{V}$ projections) attention:

$$\underset{E,C,Q,K,V}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( O_i \Big) = \underset{E,C,Q,K,V}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( C\left(E'_i\right) \Big)$$

$$= \underset{E,C,Q,K,V}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( C\big(\alpha_{ctx} \cdot \mathcal{V}E_{ctx}\big) \Big)$$

$$= \underset{E,C,Q,K,V}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( C\big(softmax(\mathcal{Q}E_{ctx} \cdot \mathcal{K}E_i^T) \cdot \mathcal{V}E_{ctx}\big) \Big)$$

with $ctx = \{w_{i-sl}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_{i+sl}\}$ the context, $i$ the target token, $O_i$ the $i^{th}$ output of the final linear layer, and $C$ a classification function[1], $E'_i$ the resulting embedding matrix from the attention mechanism, $\alpha_{ctx}$ the attention weights on the context words, $E$ the embedding matrix and $E^T$ its transposed.

[1] Often, multiple fully-connected linear layers.

But we also question the self-attention mechanism. These three projection matrices may be unnecessary in our case. So, we considered two other systems. In the first one, we removed the three projection matrices ($\mathcal{Q}$, $\mathcal{K}$ and $\mathcal{V}$) in order to refine even more our architecture, in the same spirit of the system proposed by Yang et al. [YYD+16]. We named this version `MiniBERT` **non-transforming attention** (`MiniNTA`) since input embeddings are not anymore transformed prior to the computation of the attention. In this case, the language model can be written as follows if we consider the **non-transforming** attention:

$$\underset{E,C}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( O_i \Big) = \underset{E,C}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( C\big(\alpha_{ctx} \cdot E_{ctx}\big) \Big) =$$

$$\underset{E,C}{\operatorname{argmax}} \prod_{i=1}^{n} \Big( C\big(softmax(E_{ctx} \cdot E_i^T) \cdot E_{ctx}\big) \Big)$$

The second one consists in keeping only the projection matrix $\mathcal{Q}$, to simplify the system while preserving one degree of flexibility. We named it `MiniBERT` **simple self-attention** (`MiniSSA`).

We introduce `MiniBERT` in order to adapt the complexity of the architecture to the task and dataset. It allows to deal with small corpora while benefiting from a powerful and interpretable self-attention mechanism. We expect `MiniBERT` to achieve decent performances regarding its simplicity while producing interpretable outputs. We indeed expect the only attention head to provide useful insights on the model behavior.

| Labels | Train | Test |
|---|---|---|
| Positive | 7328 | 857 |
| Negative | 13109 | 1525 |
| Neutral | 12611 | 1304 |
| MixPosNeg | 2420 | 255 |
| NA | 33448 | 3875 |
| Total: | 68916 | 7816 |

Table 1: The DEFT 2018 dataset.

# 4 Experimental setup

We evaluated our models on the two first tasks of the DEFT 2018[2] challenge. We first describe the data and then describe the systems we compare to.

**Data.** Organizers provided us a corpus of french tweets, half of them annotated according to their polarities (Table 1). A test set was provided, but no validation set, so we took 20% from the train dataset to build one.

Task 1 is a binary classification task, whose aim is to detect tweets about public transport. Polarity information have been given only on tweets about public transport, so tweets' topics can be deduced from labels given in Table 1.

Task 2 is an opinion detection task: given a tweet about public transport, the system must output the opinion expressed in it. The four possible outputs are `Positive`, `Negative`, `Neutral` and `MixPosNeg`.

**Competing approaches.** Twelve teams were competing in the contest, but we compared our models only with the five following: LIP6, IRISA, IRIT, CLaC and LIS. Each team submitted several systems, but only the one that performed the best will be presented here.

The *LIP6*' system is a 2-layer GRU network with attention applied on the output of each GRU layer. The attention is computed by calculating the distance between GRU's outputs and an attention vector $\boldsymbol{a}$, followed by a softmax operator. Here, the attention vector can be seen as a filter preventing words not conveying sentiments or opinions to alter the resulting embeddings. The *IRISA*'s system is a 2-layer biLSTM followed by a softmax layer. They used fastText embeddings, pre-trained on a large scale corpus, as input to their system. Finally, the *IRIT*'s system is probably the bigger one. They chose to train their own word embeddings and use fastText embeddings conjointly. So, the input of their system is the concatenation of both kinds of embeddings. Then, the input is processed in parallel by three subsystems: a 1D convolutional network, a biGRU network and a system computing an average embedding followed by a relu layer. All three outputs are finally concatenated and fed to a classification layer. In the *CLaC* system, the corpus is cleaned and encoded as bags of words before being classified using SVM. Finally, the *LIS* system rely on two sets of seed words often used, in the positive and negative tweets of the corpus. They used `Word2vec` embeddings pre-trained on a french corpus of tweets to compute the distance between the words in a tweet and the seed words to classify the tweets.

**Our approaches.** The main advantage of `MiniBERT` over its competitors is its simplicity, making it easy to train. Indeed, `MiniBERT` can be trained quickly, not only because of its reduced number of trainable parameters, but also because multiple input sequences can be processed simultaneously in one step, while LSTM-based models require to read their inputs one token at a time (no parallelization). We also did not rely on additional data other than the DEFT corpus, mainly because our purpose was to evaluate `MiniBERT` on its own. We conducted multiple experiments with different setups. We followed the same training strategy as `BERT`, that is to say `MiniBERT` was first pre-trained on the MLM task, using only the DEFT corpus, then fine-tuned on tasks one and two. For both tasks, we replaced the MLM head by a classification head taking as input the average embedding outputted by the attention head, which is then fed to a 2-layers fully connected network predicting the sentences' labels. The embeddings' size is set to 32, bigger embeddings did not lead to better results. Finally, we lemmatized the corpus since we supposed that grammatical nuances are not relevant in this context. Experiments with the raw corpus obtain similar results.

# 5 Results

Performances (Micro-Fscore) of all `MiniBERT`[3] variants, `CamemBERT` as well as the systems described in Section 4, are presented in Table 2. We used the `CamemBERT-Base` model available on PyTorch, and we fine-tuned it on both Task 1 and Task 2.

On Task 1, all variants of `MiniBERT` perform on-par with the best models submitted to the competition. It

[2] `https://deft.limsi.fr/2018`

[3] We kept the best performing model, amongst all our experiments, for each `MiniBERT` variant.


(a) `MiniSA`. (b) `MiniSSA`. (c) `MiniNTA`.

Figure 2: Attention matrices extracted from all `MiniBERT` variants, fine-tuned on task 2.

seems that this task is too simple to discriminate between systems. However, it shows that `CamemBERT` does not achieve decent performances (both on raw and lemmatized tweets). These first results suggest that, huge and generic models trained on mainstream documents could be irrelevant to solve domain-specific tasks where few data is available.

The very same trend can be observed on Task 2. `CamemBERT` is clearly outperformed by all the other models. `MiniBERT` systems are way ahead the systems based on simpler classifiers but a bit behind the best models submitted to the competition[4]. While we acknowledge that increasing performances is a genuine objective, the lower performances of `MiniBERT` may be compensated by its interpretability and simplicity, providing a powerful and light alternative to big attention-based models.

**`MiniBERT` is explainable.** `MiniBERT` was not intended to out-perform its competitors, since it is a simple architecture, and we trained it without relying on external data (such as pre-trained embeddings). We focused on providing an easily trainable and deployable model, but also easy-to-interpret. Attention is a mechanism easy to grasp, but not within `BERT`, since attention heads are interconnected to each others, making hard to explain how, and why, its outputs are computed. This is why we propose `MiniBERT`, the smallest `BERT`-based architecture. Because it consists in a single attention head, it is easy to understand how its outputs are computed. We suppose that the way attention is distributed could reveals insights on the model behavior. As an example, let's focus on the attention matrix computed by `MiniSA` fine-tuned on Task 2, shown in Figure 2a. Rows indicate how the attention of each word is distributed across the whole sentence. The attention matrix indicates that the most important word to classify the opinion of the tweet is "puer" (to stink), which, indeed, tends to express a negative opinion. The same model fine-tuned on Task 1 attends mostly on the word "bus", which is relevant to classify the tweet as related to public transport. This behavior is even more remarkable in the case of `MiniSSA`, the mechanism we introduce, as shown in Figure 2b. While being three times smaller (only $\mathcal{Q}$ to learn) than `MiniSA`, `MiniSSA` obtained very similar results and its attention is less scattered, leading to more legible attention matrix, and thus more interpretable outputs.

On the other hand, `MiniNTA` is behaving differently.

| Model | Task 1 | Task2 |
|---|---|---|
| `MiniSA` | 0.83 | 0.63 |
| `MiniSSA` | 0.83 | 0.64 |
| `MiniNTA` | 0.83 | 0.62 |
| LIP6 | 0.83 | 0.66 |
| IRISA | 0.83 | 0.70 |
| IRIT | 0.82 | 0.70 |
| CLaC | 0.78 | 0.34 |
| LIS | N/A | 0.48 |
| `CamemBERT` | 0.67 | 0.39 |

Table 2: Performances of `MiniBERT` and other systems.

[4] `MiniBERT` would have been ranked 7, behind the LIP6 system, on Task 2.

| Attention | Task 1 | Task2 |
|---|---|---|
| *without pre-training* | | |
| MiniSA | 0.83 | 0.63 |
| MiniSSA | 0.83 | 0.64 |
| MiniNTA | 0.83 | 0.62 |
| *with frozen embedding/attention layers* | | |
| MiniSA | 0.71 | 0.40 |
| MiniSSA | 0.70 | 0.42 |
| MiniNTA | 0.67 | 0.41 |

Table 3: Evaluation of pre-training in MiniBERT

| MLM? | L | H | Task 1 | Task 2 |
|---|---|---|---|---|
| No | 1 | 2 | 0.40 | 0.36 |
| Yes | 1 | 2 | 0.82 | 0.57 |
| No | 2 | 1 | 0.48 | 0.39 |
| Yes | 2 | 1 | 0.82 | 0.60 |

Table 4: Performances of MiniSA with $L$ layers and $H$ attention heads per layer. Models are either pre-trained on the MLM task, or directly trained on the target task.

Words attend only on themselves, as can be seen in Figure 2c, which makes MiniNTA more similar to Word2vec. But, since performances of MiniNTA are similar to the two others variants, this may suggest that attention, more specifically contextual information, is not required to solve those tasks. However, we think that attention is what enables explainability of the MiniBERT architecture. So, we investigated on what could prevent MiniNTA to capture attention. We replaced the softmax function with the Taylor-softmax [dBV15], which is basically a smoothed softmax operator. This resulted in less crisp, or more distributed, attention. Still, the attention given to other words remains too low, and this model followed the same trend as MiniNTA.

**Is pre-training required?** So far, MiniBERT was first pre-trained on the MLM task, then fine-tuned on the target tasks. In the case of BERT, pre-training aims to produce a generic model capturing many syntactic and semantic features that can be reused to solve downstream NLP tasks. Specializing this generic model is done by transferring BERT's knowledge in the new system through fine-tuning. But we suppose that, in this simple task, many syntactic features are not needed. Hence, pre-training MiniBERT models may not be necessary. We actually observe that fine-tuning MiniBERT alters greatly the pre-trained attention parameters.

To verify further that pre-training is not required in this context, we trained the same MiniBERT models, from scratch, directly on Task 1 and Task 2. To check the relevance of knowledge gathered through pre-training, we also fine-tuned MiniBERT after freezing both the embedding and attention layers. Results shown in Table 3 indicate that, indeed, latent features extracted by the MiniBERT's attention head pre-trained on the MLM task (the frozen models) are not sufficient to solve Task 1 nor Task 2. Furthermore, models trained directly on the target task tend to perform better, suggesting that, in this particularly simple case, pre-training is counter productive, while being more expensive. However, our experiments also show that pre-training seems to be mandatory when the model contains more than one attention heads, no matter what kind of attention is employed. Indeed, we trained MiniBERT models with two attention heads, instead of only one, and we observe in Table 4 that their performances were significantly degraded if they were not pre-trained on the MLM task first.

# 6 Conclusion

We considered adapting the self-attention mechanism introduced by Devlin et al. [DCLT18] to the context of small domain specific corpora. To this aim, we propose MiniBERT, a tiny architecture that refines the BERT mechanisms in this context. We derived three models from this proposal: MiniSA, MiniSSA and MiniNTA. MiniSA consists in a single self-attention head and the two others are attempts at simplifying the attention mechanism, to improve explainability [Rud19]. In particular, the *simple self-attention* mechanism we introduce, as part of MiniSSA, has the double benefits of improving explainability and reducing, by a factor of 3, the number of learnable parameters. Furthermore, experiments showed that it did not compromise accuracy.

The reduced number of parameters allows MiniBERT models to be trained quickly. Besides, when BERT requires to be pre-trained on large scale corpora, our simple architecture can be trained directly on a small corpus without requiring external data. Actually, our experiments show that larger models have difficulties to specialize to the DEFT corpus, while MiniBERT's small set of parameters is easily trained.

We showed experimentally that, despite its **very**

**small size and low footprint**, `MiniBERT` obtains results almost on par with its competitors. Even though `MiniBERT` does not outperform them, its losses are compensated by the simplicity of its training and the interpretability of its outputs. Indeed, observing the attention matrix produced by the model gives precious insights on the model behaviour. In particular, most important words are clearly highlighted, which is important to ensure that the model is behaving as expected. For example, in a sentiment analysis task, it would be easy to detect if the model is gender biased by analyzing if attention is put on gender specific words.

Of course, a lot of work remain to be done. For example, it is not clear what are doing the $\mathcal{Q}$, $\mathcal{K}$ and $\mathcal{V}$ matrices, which explains our attempt at removing them (`MiniNTA`). We showed that one of them is at least required to compute properly the attention (`MiniSSA`), thus we do not plan anymore to remove them, but to replace them by more graspable transformations. For example, replacing these matrices by a set of constrained matrices, such as rotation, translation or scaling matrices, may not degrade too much the expressiveness of the model while helping us understand better what is actually done. Future work will also be conducted to assess `MiniBERT`'s capacity to scale up to harder problems on a variety of domain-specific corpus.

# References


[CKLM19] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What Does BERT Look At? An Analysis of BERT's Attention. *arXiv:1906.04341 [cs]*, June 2019.

[dBV15] Alexandre de Brébisson and Pascal Vincent. An exploration of softmax alternatives belonging to the spherical loss family. *arXiv preprint arXiv:1511.05042*, 2015.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[dWP20] Adrian de Wynter and Daniel J Perry. Optimal subarchitecture extraction for bert. *arXiv preprint arXiv:2010.10499*, 2020.

[GDA20] Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*, 2020.

[HSMN12] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882, 2012.

[LG14] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Neurips*, 27:2177–2185, 2014.

[LGD15] Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225, 2015.

[LJ15] Jiwei Li and Dan Jurafsky. Do multi-sense embeddings improve natural language understanding? *arXiv preprint arXiv:1506.01070*, 2015.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[MH09] Andriy Mnih and Geoffrey E Hinton. A scalable hierarchical distributed language model. In *Neurips*, pages 1081–1088. Citeseer, 2009.

[MLFR05] Irina Matveeva, G Levow, Ayman Farahat, and Christian Royer. Generalized latent semantic analysis for term representation. In *Proc. of RANLP*, page 149, 2005.

[PGB+18] Patrick Paroubek, Cyril Grouin, Patrice Bellot, Vincent Claveau, Iris Eshkol-Taravella, Amel Fraisse, Agata Jackiewicz, Jihen Karoui, Laura Monceaux, and Juan-Manuel Torres-Moreno. Deft2018: recherche d'information et analyse de sentiments dans des tweets concernant les transports en île de france. In *DEFT 2018*, volume 2, pages 1–11, 2018.

[Rud19] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[TDB+14] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pages 151–160, 2014.

[VM14] Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[YYD+16] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *NAACL*, pages 1480–1489, 2016.