TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection

Lecture d'Articles et de Documents Scientifiques

Clément GUERIN and Maxime LABORDE

1 Introduction

With the fact that more data are now available than ever before, major companies have now the resources to develop a business offer or create a service using these data. Nowadays, one of the main problematic is to be able to deal with language tasks. An example of application of these tasks are the virtual assistants which, being given a question or an assertion, are able to give an answer or to act in consequence, which implicates that they "understand" what they are told. These tasks are the so-called NLP tasks, this acronym stands for Natural Language Processing. The possible ways to orientate the research to better understand the data are various but in the "TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection" paper [**1**], the problem has been limited to QA ("Question and answer") tasks and more specifically to AS2. The AS2 consists in selecting the correct answer to a question from a given set of answer candidates. Such a difficult task requires a huge amount of data examples, which are now accessible, to be able to train an algorithm. In the recent years, a method has emerged with excellent results compared to previous ones to solve the NLP tasks: BERT. The method created in the TANDA paper goes beyond the BERT algorithm but it is essential to understand the way it works first in order to discuss the extra steps it has been added. To be able to understand BERT we first need to discuss about word embedding and transformers. Certain sections will be illustrated with a Google Colab, using our own code.

2 Context and related work

2.1 Word embedding

Traditionally, to represent documents and words, huge matrix having as many rows as there are documents and as many columns as there are different words in these documents were used. The matrix M is filled with 1 at the position $M_{i,j}$ if the i-th document contains the j-th word and 0 otherwise. The problem is that if the number of documents and words to deal with are too important, representing it by a matrix in a classical way is almost impossible. To be able to deal with this curse of dimensionality the word embedding were used. It consists in representing each word in a vector space with a dimension a lot smaller than in the previous method. Words which are close in meaning should, in this smaller space, occupy close positions in terms of vector distance. Therefore, the cosine of the angle between words is calculated and the closer it is to 0, the closer the words are in terms of meaning. This allows to represent words using smaller objects. As an example let's consider that we have 10^6 documents containing a total of 10^6 different words. The basic matrix to represent it would contain 10^{12} elements. Now, using word embedding, we assume that we can represent each of these 10^6 words in vectors of dimension 100 which is one of the most common size used. The dimensions of the objects to deal with drop to 10^8, which is a massive complexity improvement. Here is a fictional example about how words, given a certain corpus, could be represented in a two-dimensional space. "cat" and "kitten" which are extremely close semantically will be extremely close in terms of vector distance. "dog" is fairly close to "cat" and "kitten" but far from "house" which is totally different semantically speaking [**2**]. Illustration of word embedding on Google Colab (section 1): **Guerin Laborde Colab**

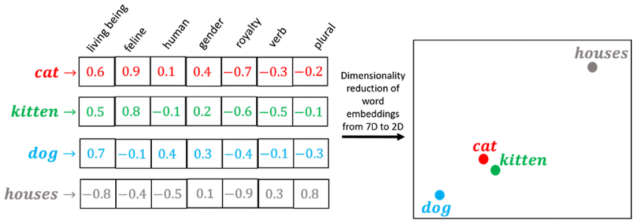

Figure 1: Adapted from [**2**], 2D word embedding

2.2 Transformers

Transformers for their part are objects designed to be able to handle natural language data and are able to process the data in any particular order. This specificity makes it able to use parallel computations, allowing a huge gain in performance. Transformers are composed of an Encoder consisting of many encoding layers and a Decoder composed of many decoding layers. First the transformer takes as an input a sentence and more specifically the embeddings of this sentence which are given to the first encoding layer. The results are then propagated through the other encoders.

A general description of an encoder is the following : Given an input data X_i, a certain function will be applied to it alongside adding a bias (i.e : $X_i \rightarrow \sigma(WX_i + b)$). Depending on the dimensions of the input and of the matrix W representing weights, the computed output will have a new dimension. This process is repeated with different weight matrix and bias the same number of time than the number of encoding layers in the network. For transformers, many

other layers are added like Multi-Head attention layers and Add & Norm layers.

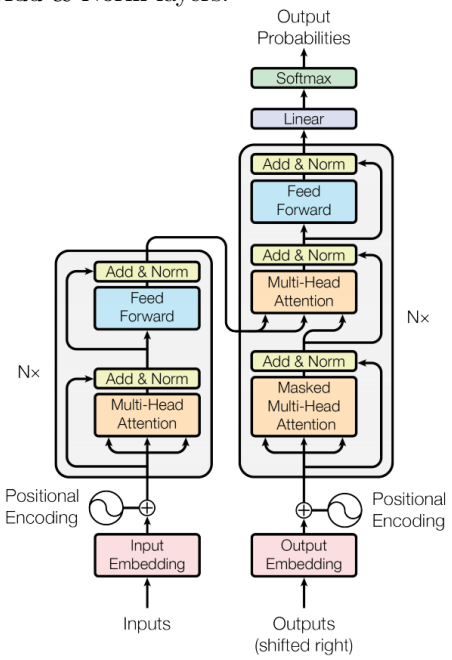

Figure 2: From [**3**], Transformer architecture

Multi-Head attention layers : An attention layer takes as an input the word embedding of a sentence. This layer consist simply in computing a weighted average of the different vectors composing the input. The size of the output produced is the same as the input. A multi-head attention layer, then, consists simply in computing multiple attention layer in parallel then concatenating the results. This technique allows to determine context, importance and try to associate words having a similar meaning in a sentence.

Add & Norm layers : these layers are used to standardize the range of values the other layers will have to deal with, diminishing the variability of the data and allowing a better generalization.

Then the decoding part is processed using similar layers. At the top of it some linear and non-linear function (e.g: softmax) can be applied to obtain probabilities which we will discuss more in details in the TANDA description.

2.3 BERT

Now that all the above objects have been explained we will now describe the functioning of BERT which stands for "Bidirectional Encoder Representations from Transformers" and relies on transformers. The notable difference is now that only the encoding part of the transformers are used. Since the training of BERT is only accessible with a huge computing power, it now has been pre-trained in many different ways in order to be used. The pre-training consists in two tasks. In the first one, a masking mechanism is implemented. Let's consider the following sentence : "The house is huge". The masking mechanism consists in masking 15 % of the words :

- 80% of which will be replaced by the MASK token.
The house is huge → The house is MASK
- 10% of which will be replaced by a random word.
The house is huge → The house is computer
- 10% of which will not be replaced.
The house is huge → The house is huge

Then to train itself, BERT tries to predict the masked words and learn from its success and failures in predicting those missing words. All of these percentages have not been taken randomly but empirically. 15% is approximately the maximum value to guarantee that BERT has enough context to train on. Therefore, by considering this maximum value and not a lower one, BERT pre-training converges at the maximum possible speed. The (80 - 10 - 10) split between mask token, random words and not doing anything has also proven to be an empirically good solution.

The second pre-training task consists in a "next sentence prediction task". A huge dataset containing pairs of sentences are given as an input to BERT. 50% of these pairs are related (e.g :"the fire is hot" and "the fire burn") and the other 50% does not share any link (e.g :"the house is small" and "it is time to go"). BERT task consists in learning to predict correctly link between sentences. These two tasks end up in obtaining a pre-trained BERT model which is at the base of the method created in the TANDA paper. BERT is pre-trained using 16 GB of data. Figure below, shows a summary of the pre-train tasks. Illustration for using a pre-trained BERT model on Google Colab (Section 2): **Guerin Laborde Colab**

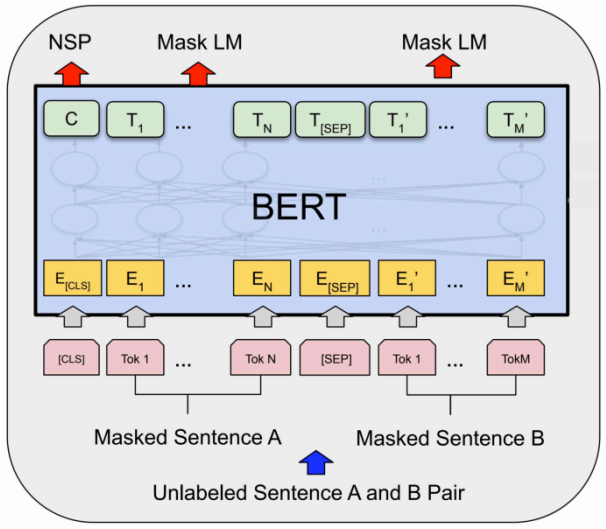

Figure 3: Adapted from [**1**], BERT pre-training phase

3 TANDA : Transfer AND Adapt

When trained on a big enough dataset, BERT has proven to give better result than any other alternative techniques. However, in order to obtain even better results an extra step of what is called "fine-tuning" is processed. This step consist in giving an additional dataset related to the task we want to calibrate BERT on, as an input. The particularity of the TANDA technique is that it processes two fine-tunings that we will describe.

3.1 First fine-tuning : Transfer

The first fine-tuning consists in "transferring" the general pre-trained model processed earlier to a specific task. In the scope of TANDA, we want to adapt BERT specifically

to AS2 tasks. Mathematically it is defined as follows : for a certain question Q_i, there is a set $S = \{s_{i,1}, ..., s_{i,n}\}$ of answer candidates. AS2 consists in selecting the answer that answers best Q_i. We note $P(Q_i, s_{i,j})$ the probability estimated by our transformer for $s_{i,j}$ to be the correct answer. The problem simply translates in selecting the index k in the set S where $k = argmax_j P(Q_i, s_{i,j})$.

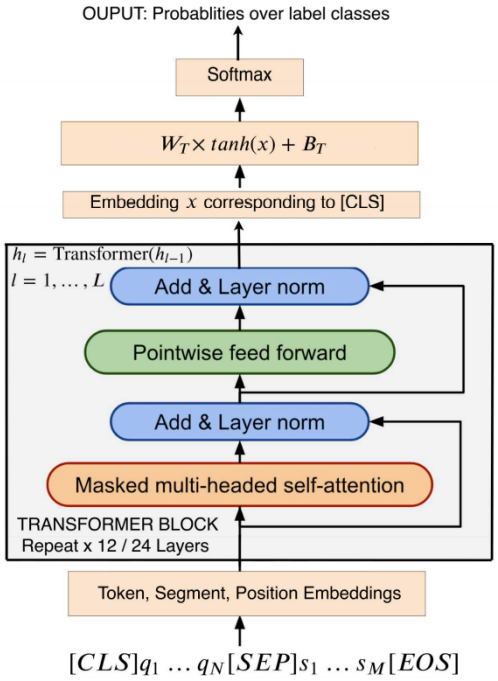

Figure 4: From [**1**], BERT architecture

The figure above shows in details how the $P(Q_i, s_{i,j})$ probabilities are estimated. First the transformer perform his task in the way defined earlier. Then, when the encoding part has been processed the encoded output is an embedding "x" which represents the relations between the text pairs considered (i.e : the question and an answer candidate). To obtain the probabilities for the answer candidates to be the correct answers we then compute $W_T * tanh(x) + B_T$. x is an embedding, W_T is a matrix representing weights used to multiply the vector x. B_T is a vector of bias.

W_T and B_T are two parameters that are going to be calibrated by the model, then a non-linearity is added through the softmax function. This function has the particularity of taking in input a vector and transform it in a vector of probabilities which sums to 1. After these steps the model has now the capacity of predicting the answer he considers to best fit the question by choosing the answer candidate with the highest probability. Illustration for using a BERT model fine-tuned for QA on Google Colab (Section 3): **Guerin Laborde Colab**

Therefore, we understand easily that the task of transferring the model to the specific AS2 task is crucial and it needs to be done with a large enough dataset. Unfortunately, AS2 is not the task for which the most dataset are available and the one that are available are relatively small (i.e about 10^4 sentence pairs) so the authors of the TANDA paper constructed a dataset named ASNQ "Answer Sentence Natural Questions". This dataset contains 57,242 distinct questions for training and 2,672 questions for the test dataset. Each candidate answers are labelled : 1, 2 and 3 referencing the answers that are not correct while 4 is used for the correct answers. The authors claim that this first fine-tuning allows the models to perform better, which we will discuss in the result part.

3.2 Second fine-tuning : Adapt

Afterwards, a second fine-tuning is performed, which will allow the model to adapt to a specific corpus (i.e a target domain). The first-tuning allowed the model to train on general question and answer tasks, meaning that it learned on a relatively large number of examples coming from various domains. The consequence is that this single fine-tuned model will produce good results for all these various domains but will perform optimally in none of them. Therefore, the interest of the TANDA technique is to feed the algorithm with a second additional dataset composed exclusively of question and answers related to a specific domain (e.g : sports, music, etc...). This second fine-tuning comes with three major advantages. First, instead of calibrating the model on a huge dataset already related to the target domain during the first fine-tuning, it allows a general first fine-tuning that can be reused. The second advantage is that since the first fine-tuning will already be processed with a huge number of answer and question set, the second fine-tuning can afford to be done on a smaller dataset. This advantage is important since finding a dataset huge enough for all the potential target domain is impossible in practice. Finally, the results the authors claimed to obtain are a significant improvement over the state of the art results.

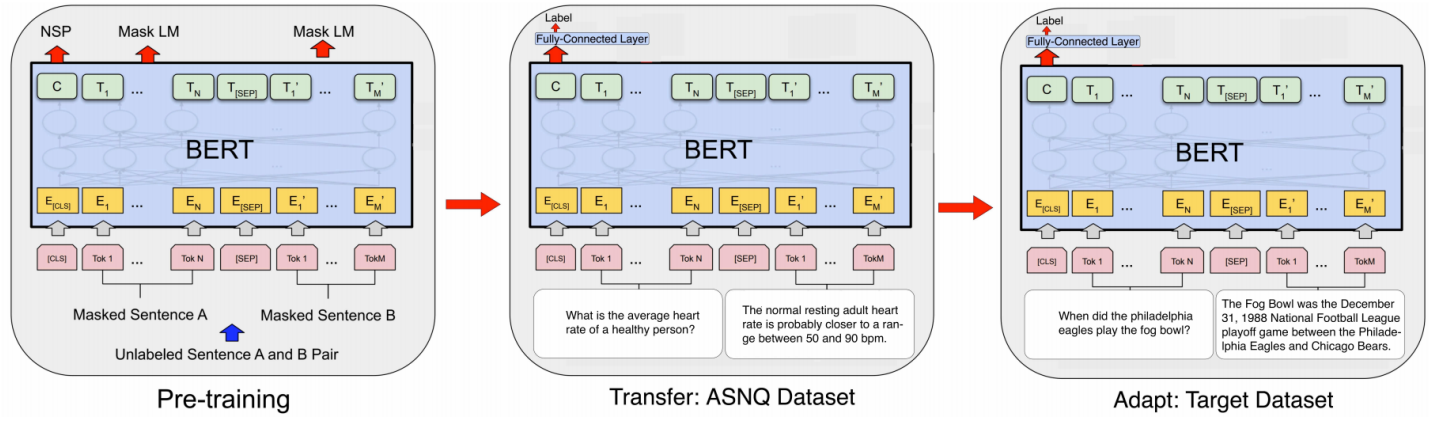

Figure 5: From [**1**], TANDA architecture showing the two distinct fine-tunings

4 Results of TANDA

In this section we are going to study the different results the TANDA method can achieve. We first need to define the different metrics that will be used to measure the different results as well as the model that were used to pre-train BERT.

4.1 Metrics used

To measure the quality of the results TANDA is capable of achieving two metrics are used : Mean Average Precision ("MAP") and Mean Reciprocal Recall ("MRR"). MAP consists simply in computing the average proportion of correct answers to the different questions, selected by the model in the test dataset. To compute MRR, it is first required to ordinate all the answer candidates by probability of being the correct answer, then a rank is attributed to each one. The MRR is then the mean of the inverse rank of the different correct answers.

4.2 Pre-trained models used

The pre-trained models used are various in order to guarantee that this new technique works in different frameworks : BERT-Base, BERT-Large, RoBERTa-Base and RoBERTa-Large-MNLI. RoBERTa ("Robustely optimized BERT approach") is an alternative to BERT which pre-trains on more data than BERT (160 GB) and without using the next sentence prediction phase. It produces even better result than BERT.

4.3 Results

To finally be able to compare the results obtained to the previous state of the art results, the authors used WikiQA and TREC-QA which are the most popular benchmarks for AS2 tasks.

WikiQA has some questions which does not have any correct answers and some which have only correct answers. As a consequence, the questions with no correct answers are not considered. Therefore, WikiQA is composed of 857 questions with an average of 10 answer candidates for each in the train dataset and 237 questions in the test dataset. Using the best possible configuration (i.e : RoBERTa-Large + TANDA (ASNQ → WikiQA) the MAP reached 92% and the MRR stands at 93.3%. These scores are impressive and redefines the state of the art for the WikiQA dataset. The previous best scores for WikiQA on the MAP was 83.4% which shows the importance of the benefits of the TANDA method.

TREC-QA is another benchmark, on which it was decided to remove questions that were without answers, only with incorrect answers and only with correct answers. It resulted in a training set composed of 1,229 questions with an average of 40 answer candidates for each question. The test set is composed of 68 questions. Using the best possible configuration (i.e : RoBERTa-Large + TANDA (ASNQ → TREC-QA) the MAP reached 94.3% and the MRR culminated at 97.4%. Once again, these score are impressive and improves by a huge margin the previous state of the art result which was 87.5% for the MAP.

5 Conclusion

TANDA is a new approach to fine-tune different BERT and transformer models in general that has proven by the quality of its results that it could be the new state of the art defining method. It consists in two different fine-tunings, the first one is made to transfer the general language model to a model adapted to a more specific task (AS2 for the purpose of this paper). It is performed using a general and large dataset. The second one consist in adapting the transferred model to a target domain. The benefits of this approach has been empirically demonstrated for the AS2 task, allowing to obtain results on two of the most used benchmarks that are huge improvements over the state of the art. In addition to that the dataset needed to adapt the model to a target domain can be a lot smaller than it was required before. ASNQ which is a large and high-quality dataset has also been developed for the purpose of this paper. This dataset could be the new standard for the AS2 task in NLP. Of course, one of the most interesting things to study for the future is whether this technique could be extended to the other NLP tasks or not.

References

[1] Siddhant Garg, Thuy Vu and Alessandro Moschitti. *TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection (2019).* `http://arxiv.org/pdf/1911.04118v2`

[2] Hariom Gautam. *Word Embedding: Basics.* `https://medium.com/@hari4om/word-embedding`

[3] Ashish Vaswani et al. *Attention Is All You Need (2017).* `https://arxiv.org/pdf/1706.03762.pdf`

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2019).* `https://arxiv.org/pdf/1810.04805.pdf`