A DIVE INTO THE PERFORMANCE ANALYSIS OF THE BERT FAMILY AGAINST THE FNC-1 TASK

Alessio Serra

January 2022

Abstract

Transformer based models have led to a revolution in NLP, as they can reach SOTA in a wide variety of tasks, and for this reason many people associate their introduction into NLP with "ImageNet moment" in Computer Vision. In this project I try to test many different kinds of Transformer encoder architecture, which belongs to the BERT family, considering both transfer learning techniques through fine tuning of pre-trained model and feature extraction to obtain contextual word embedding. All my trials were done to solve the FNC-1 challenge and led me to improve the SOTA results reported in the challenge website.

1 Introduction

The task that I will face is the Fake News Challenge (FNC-1), that was proposed in 2017 to find a countermeasure to the spread of false information, probably one of the most serious challenges facing the news industry today.
Due to the difficulty in evaluating the truthfulness of a news, FNC has divided the problem into a series of simpler steps, the first of which is the goal of the challenge. The latter is about Stance Detection, which is the process of understanding what the news is writing about a topic. In particular, the model to be submitted will estimate the position of a body of the text of a news article with respect to a headline, deciding whether the text agrees, disagrees, discusses or is not related to it. To solve the problem I followed two distinct approaches:

- **Fine tuning.** I tried to fine tune several pre-trained models, which belong to the "BERT family", but that differ in the way or in the dataset in which they are pre-trained, in the number of levels and parameters. In particular I tested the BERTbase and BERTlarge model proposed in the original paper, a smaller version of it, called "small Bert", Electra, Albert, "Bert with talking heads" and also a "Bert Expert" model, which is pre-trained on a textual entailment task.

- **Contextual word embeddings.** All previous models are also used to extract contextual word embeddings, which are given as input to a Convolutional neural network.

For both parts I reported the results obtained, together with all the intermediate experiments carried out to improve performance and for hyperparameters tuning.

2 Dataset

The dataset, made publicly available on GitHub by the organizers, consists of a pair of news article titles and text snippets, that is obtained associating each title with its article or with another one. All texts are derived from the Emergent dataset for Stance's classification, created by journalist Craig Silverman. It is all contained into four csv file:

- *trainBodies, competitionTestBodies:* contain the body text of articles with corresponding IDs.
- *trainStances, competitionTestStances:* contain the labeled stances for pairs of article headlines and article bodies for the training set.

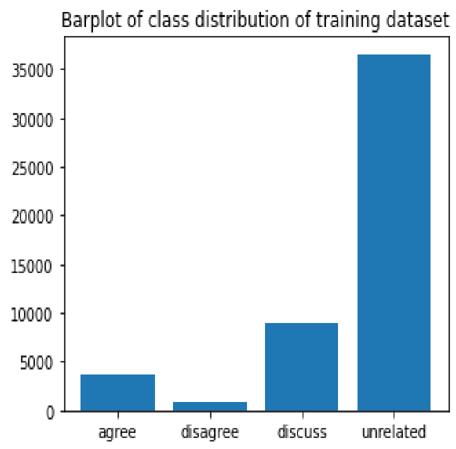
Headline	Robert Plant Ripped up $800M Led Zeppelin Reunion Contract
agree	Led Zeppelin's Plant turned down £500 MILLION to reform supergroup
disagree	No, Plant did not rip up an $800 million deal to get Zeppelin together
discuss	Plant reportedly tore up an $800 million Led Zeppelin reunion deal
unrelated	Richard Branson's Virgin Galactic is set to launch SpaceShipTwo today

Table 1: Example of headline and article bodies with relative Stance provided by challenge organizers.

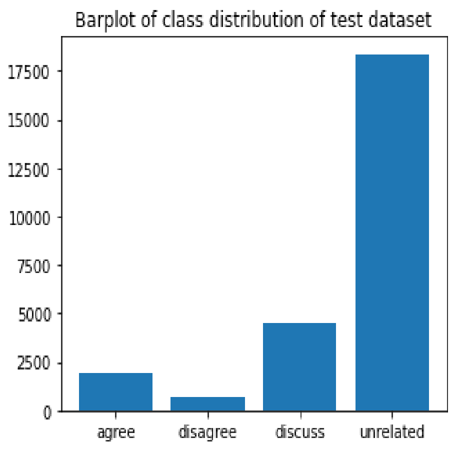
The aim of the network is to solve a textual entailment task, in which the model has to classify each pair *"Headline", "articleBody"* given as input with one of the four possible labels, namely *"agree", "disagree", "discuss"*, and *"unrelated"*. In *Table 1* is possible to see 4 samples from the dataset.

Dataset distribution and composition. In the training set, the number of unique headline are equals to 1'648 and the number of unique bodies are 1'683, but the total size of the dataset is 49'972, as the same headline is combined with different bodies. This results in a high class imbalance, where most of the samples (73%) belongs to the "unrelated" class, as we can see in *Figure 1.*

The test set provided has almost the same distribution of classes, but it contains 894 unique headlines, 904 unique bodies and 25'413 samples, so that's almost the 51% of the training set.



(a) Plot of training set distribution.



(b) Plot of test set distribution.

Stance	**Cardinality**	**Percentage**
unrelated	36545	73.1
discuss	8909	17.8
agree	3678	7.4
disagree	840	1.7

Table 2: Distribution of classes on training set

Stance	**Cardinality**	**Percentage**
unrelated	18349	72.2
discuss	4464	17.6
agree	1903	7.5
disagree	697	2.7

Table 3: Distribution of classes on test set

3 Evaluation Metrics

The evaluation metrics used in this project are of two types, the FNC score, which was proposed in the official FNC-1 challenge, and other metrics related to the F1 computation.

FNC score. This metric assigns 0.25 points if an article is correctly classified as related or "unrelated" to a given headline. Then if it is related, additional 0.75 points are assigned if the model correctly distinguish among the 3 related class, so *"agree"*, *"disagree"*, or *"discuss"*.
This is done to account for the unbalance of the classes, as shown in Figure 2, and because the related/unrelated classification task is much simpler and less relevant for the detection of fake news. A visual schema is provided in Figure3.

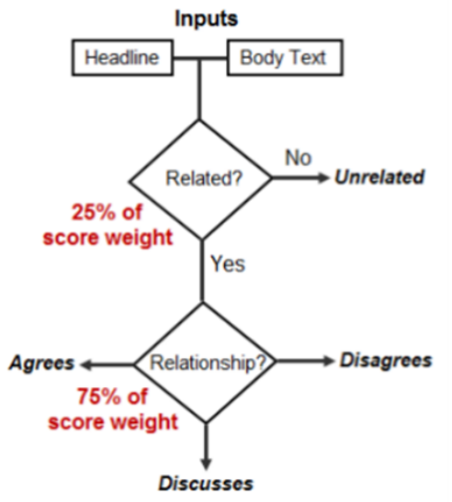


Figure 2: Visual schema of FNC score computation.

F1 metrics. This metric is inspired by the paper (Hanselowski et al., 2018), where the authors criticize the FNC score, as not sufficient to cope with the imbalance of the classes. In fact, for example, a classifier specialized in distinguishing well only the *"unrelated"* class, would still be able to obtain a very high score, as he would get 0.25 for each *"unrelated"* sample and at least another 0.25 for any other related sample, even if not correctly classified. More over, if he chose to assign the *"discuss"* class for each related sample, he would guess 63% of the time, as there is an additional internal imbalance to the related classes. For this reason it is better to also use the F1 score for each class and their average, so the F1 macro-average score, which can provide a summary of performance, giving more importance to small classes.

4 Model architecture

In all the experiments carried out I make extensive use of transfer learning techniques starting from pre-trained models belonging to the BERT family, available on TensorFlow hub. Before moving on to the analysis of the architectures and the results obtained, below I present a brief description of all the models used, emphasizing the differences compared to the original BERT model, in order to better understand the results obtained.

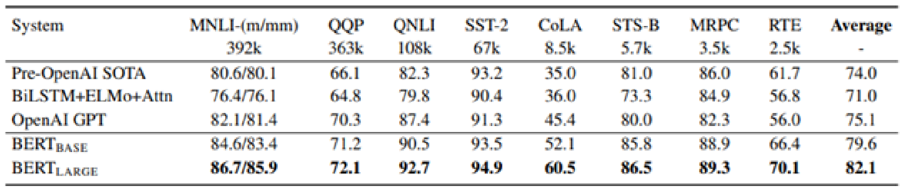
4.1 Original BERT

All the pre-trained networks used share the architectural models of the Transformer encoder and belong to the Bert family which, on the basis of self-attention, analyses the entire context of each word, taking into account the tokens before and after it. From this characteristic derives the name "**BERT**: **B**idirectional **E**ncoder **R**epresentation from **T**ransformers".
The father of this family is the BERTbase model, introduced with *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* (Devlin et al., 2018) which presents an architecture composed of a stack of L encoders, a hidden dimension H and number of attention heads A.
Each attention head expands the model's ability to focus on different properties of the input, while each level performs a multi-head attention calculation on the word representation of the previous level to create a new intermediate representation of the same size, which is equal to the size of the hidden representation H.
The authors proposed different configurations of the model, modifying the number of parameters L, H and A and noting that the increase in the size of the model also corresponds to an increase in performance, even if this results in a slower and more resource-intensive model.

In this project I use two configurations of BERT, that are BERTlarge with L=24, H=1024, A=16 and 340M of parameters and BERTbase with L=12, H=768, A=12 and 110M of parameters.
These huge models have been trained in a self-supervised way on Wikipedia and BookCorpus datasets using language modeling, specifically Masked Language Model (MLM), useful for learning bidirectional representation of input, and next sentence prediction (NSP), useful for capturing the relationship between pairs of input sentences. The latter is probably the main reason why the BERT family models can perform well in language inference task.

4.2 BERT expert

If we start from a pre-trained BERT model that is then fine-tuned on the downstream task, we can obtain impressive results. However, the latter can be further improved starting from a BERT model that aligns or transfers better to the task, particularly when you have a low number of samples; To do this is needed a



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Figure 3: BERT performances on GLUE tasks, notice that it can reach very good accuracy on MNLI task.

BERT model trained on a similar domain and/or on similar task.
Starting from this assumption I have tried to fine tune using a BERTbase model first trained, as always, on Wikipedia + BooksCorpus on MLM task and then further fine-tuned on MNLI [1], a dataset for multi-genre natural language inference, for 12 epochs with a batch size of 32 and Adam optimizer using a 1e-5 learning rate.
In this way, the model has already been trained on textual entailment for many epochs and with much more training data, around 433k samples. The domain is also quite similar, since that are used both spoken and written text, although in this case the premise and the Hypothesis are almost the same length, while in FNC the body text is much larger respect to the headline, in fact they have a mean size of 11 and 370 tokens respectively. Another difference is in the number of classes, as MNLI consider that the premise can be neutral, contradiction or entailed in the hypothesis.

4.3 Small BERT

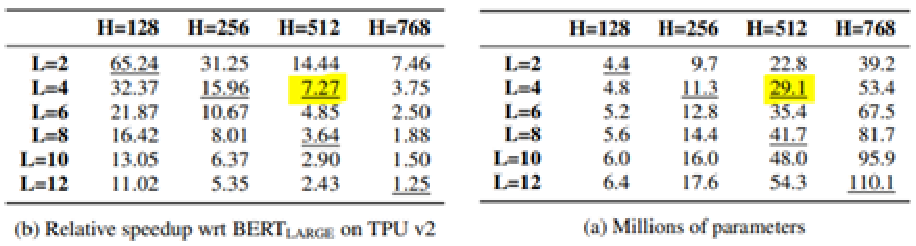
Small Bert was proposed in *"Well-Read Students Learn Better: On the Importance of Pre-training Compact Models"*, (Turc et al, 2019) as a very compact and fast variant of BERT, which is still able to achieve excellent performance on many tasks.

Difference from previous model. The main novelty of this model is the combination of the standard two-staged training approach used in BERT with the technique for model compression proposed by (Hinton et al., 2015), i.e. *knowledge distillation*. In the latter, the smaller student model is trained to recover the predictions of a highly accurate teacher using as ground truth, not only the hard labels of the dataset, but also soft labels related to the teacher's predictions, which allows better generalization than the gold hard labels.
The soft labels coincide with the class probabilities produced by the teacher, obtained with a Softmax on the logits divided by a temperature parameter, even if the latter is often kept equal to 1 to get better performances, as pointed in *"BERT2DNN: BERT Distillation with Massive Unlabeled Data for Online E-Commerce Search"* (Jiang et al, 2020).

[1]The dataset is available at https://cims.nyu.edu/ sbowman/multinli/.

This mixed approach is called *Pre-trained Distillation*, where are executed in sequence three training operations: masked language model, task-specific distillation and optional fine-tuning.

Performance. It has the same BERT architecture but fewer and smaller Transformer blocks and number of attention heads, in fact the model used in this project has L=4, H=512 and A=8, thus resulting in 29.1M parameters against the 110M of BERTbase.
This model is very good to explore trade-offs between speed, size and quality, in fact the small Bert used obtains a speed-up respect to BERTlarge of a 7.27 factor reducing the accuracy only from 86.7% to 82.8% on the MNLI task of GLUE and reducing the number of parameters from 334M to 29.1M.



	H=128	H=256	H=512	H=768
L=2	65.24	31.25	14.44	7.46
L=4	32.37	15.96	7.27	3.75
L=6	21.87	10.67	4.85	2.50
L=8	16.42	8.01	3.64	1.88
L=10	13.05	6.37	2.90	1.50
L=12	11.02	5.35	2.43	1.25

(b) Relative speedup wrt $\text{BERT}_{\text{LARGE}}$ on TPU v2

	H=128	H=256	H=512	H=768
L=2	4.4	9.7	22.8	39.2
L=4	4.8	11.3	29.1	53.4
L=6	5.2	12.8	35.4	67.5
L=8	5.6	14.4	41.7	81.7
L=10	6.0	16.0	48.0	95.9
L=12	6.4	17.6	54.3	110.1

(a) Millions of parameters

Figure 4: Comparisons of different configuration of small BERT respect to BERTlarge regarding the speed up and the number of parameters. In yellow are highlighted the statistics of the small-BERT model chosen in the project.

4.4 Albert

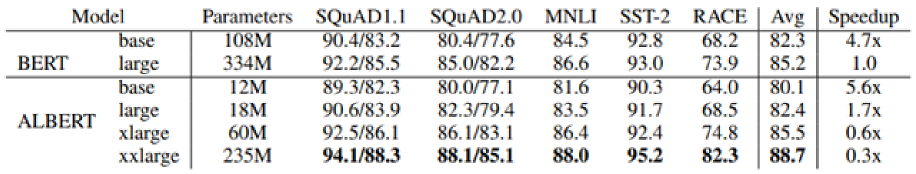
Albert is "A Lite" version of BERT with greatly reduced number of parameters, that was originally presented in *"ALBERT: A Lite BERT for Self-supervised Learning of Language Representations"* (Lan et al, 2019).
They introduced three main contributions to BERT's design choices, i.e. *Factorized embedding parameterization, Cross-layer parameter sharing, Inter-sentence coherence loss.* The latter allow for a reduction in parameters, lower memory consumption and a slight increase in training speed, thus obtaining a model that scales much better than the original Bert.

Factorized embedding parameterization. The one-hot vector of each word is not projected directly into the hidden space, which in BERTlarge is 1024, but is first projected into a lower embedding space of dimension E. This results in a reduction in complexity, ranging from $O(V \times H)$ to $O\ (V \times E + E \times H)$, and a significant reduction in the number of embedding parameters, which is much more evident if H>>E. Notice that V is equal to the size of the vocabulary, which is usually composed of about 30'000 elements.

Cross-layer parameter sharing. All parameters between transformer layers are shared, in particular both feed-forward network (FFN) and sharing attention parameters.

Inter-sentence coherence loss. The Next Sentence Prediction task on which BERT is pre-trained is modified, in fact (Lan et al, 2019) poses a critique, stating that the network learns more about topic prediction than coherence prediction. This is a limit as the former is not only easier to solve, but also overlaps more with what is learned using the Masked LM loss.
In the revised version of NSP, the same BERT technique is used to generate positive examples (two consecutive segments of the same document), while for the negative examples are used the same two consecutive segments but with their order swapped. In this way the model is forced to learn more about coherence properties and, as a result, Albert models consistently improve the performance on multi-sentence encoding tasks.

Performance. Thanks to the changes introduced, the Albert model has a drastic reduction in the number of parameters, going from 334M of the BERTlarge to 12M, but the speedup is only slightly improved by a factor 5.6x compared to the BERTlarge model.
Regarding performances on the MNLI task of GLUE, they are passed from 86.7% to 81.6%.



Model		Parameters	SQuAD1.1	SQuAD2.0	MNLI	SST-2	RACE	Avg	Speedup
BERT	base	108M	90.4/83.2	80.4/77.6	84.5	92.8	68.2	82.3	4.7x
	large	334M	92.2/85.5	85.0/82.2	86.6	93.0	73.9	85.2	1.0
ALBERT	base	12M	89.3/82.3	80.0/77.1	81.6	90.3	64.0	80.1	5.6x
	large	18M	90.6/83.9	82.3/79.4	83.5	91.7	68.5	82.4	1.7x
	xlarge	60M	92.5/86.1	86.1/83.1	86.4	92.4	74.8	85.5	0.6x
	xxlarge	235M	**94.1/88.3**	**88.1/85.1**	**88.0**	**95.2**	**82.3**	**88.7**	0.3x

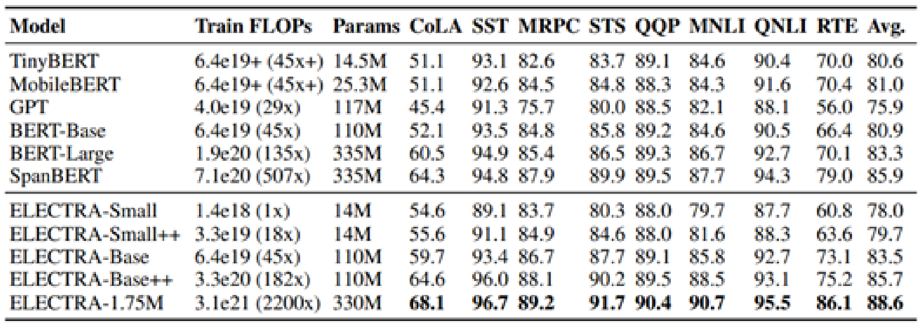
Figure 5: Comparison of performances on different downstream tasks of different Albert's configuration and original BERT models. Notice that the model used in the experiments is Albert$_{base}$.

4.5 Electra

Electra was first presented in *"ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators"* (Clark et al, 2020) and mainly proposed a more efficient pretraining task, i.e. *replaced token detection*, modifying the Masked LM.

Replaced token detection. This task, like MLM, randomly selects words in the text, but unlike the latter it does not mask them with a token, but replaces them with plausible alternatives returned by a small generator network, which samples from a proposal distribution. So the model must behave as a

modified word discriminator, not as a *masked word generator*.
However this is not an adversarial model, in fact the generator is trained with maximum likelihood and not to fool the discriminator, due to the difficulty of applying GANs with text, as pointed in (Caccia et al., 2018).
This change has led to a better contextual representation of the inputs, as the task is defined on all input tokens and not just on 15% of masked words. Furthermore the network sees the same type of input both during training and during fine-tuning, while with the original Masked LM, the network sees the artificial [MASK] tokens.
The gains in performances mainly regard the speed-up in the training and in the reduction of parameters. These reflects in higher accuracy, considering the same capacity of the network and training time, in fact it is able to reach almost 5 points more on the average of GLUE respect to a same-size model, in particular BERT-small, that can reach 75.1% against the 79.9% on GLUE.
In this project has been used ELECTRA-small, that can reach 79.7% on MNLI of GLUE respect to the 87.7% of BERT large.



Model	Train FLOPs	Params	CoLA	SST	MRPC	STS	QQP	MNLI	QNLI	RTE	Avg.
TinyBERT	6.4e19+ (45x+)	14.5M	51.1	93.1	82.6	83.7	89.1	84.6	90.4	70.0	80.6
MobileBERT	6.4e19+ (45x+)	25.3M	51.1	92.6	84.5	84.8	88.3	84.3	91.6	70.4	81.0
GPT	4.0e19 (29x)	117M	45.4	91.3	75.7	80.0	88.5	82.1	88.1	56.0	75.9
BERT-Base	6.4e19 (45x)	110M	52.1	93.5	84.8	85.8	89.2	84.6	90.5	66.4	80.9
BERT-Large	1.9e20 (135x)	335M	60.5	94.9	85.4	86.5	89.3	86.7	92.7	70.1	83.3
SpanBERT	7.1e20 (507x)	335M	64.3	94.8	87.9	89.9	89.5	87.7	94.3	79.0	85.9
ELECTRA-Small	1.4e18 (1x)	14M	54.6	89.1	83.7	80.3	88.0	79.7	87.7	60.8	78.0
ELECTRA-Small++	3.3e19 (18x)	14M	55.6	91.1	84.9	84.6	88.0	81.6	88.3	63.6	79.7
ELECTRA-Base	6.4e19 (45x)	110M	59.7	93.4	86.7	87.7	89.1	85.8	92.7	73.1	83.5
ELECTRA-Base++	3.3e20 (182x)	110M	64.6	96.0	88.1	90.2	89.5	88.5	93.1	75.2	85.7
ELECTRA-1.75M	3.1e21 (2200x)	330M	**68.1**	**96.7**	**89.2**	**91.7**	**90.4**	**90.7**	**95.5**	**86.1**	**88.6**

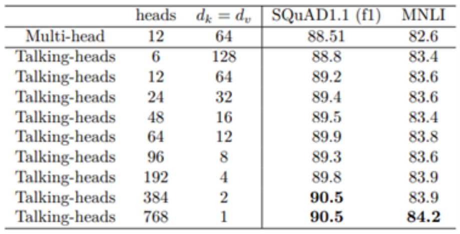
Figure 6: Results of Electra models on the GLUE test set.

4.6 BERT with Talking Heads

BERT with Talking-Heads Attention is a Transformer Encoder for text that modifies the BERT architecture by using talking-heads attention, instead of multi-head attention, and by using a gated linear unit with GELU activation as the first layer of the position-wise feed-forward networks, instead of an ordinary dense layer, as proposed by *"Talking-Heads Attention"* (Shazeer et al, 2020) and *"GLU Variants Improve Transformer"* (Shazeer 2020).

Original multi-head attention. In the traditional "multi-head" attention, the input, which can be an embedding in the first layer or the output of the previous layer in the following ones, is multiplied by three weight matrices, obtaining a Query, a Key and a Value matrix, which are then used to calculate one head, i.e. the attention matrix. In order for the model to focus on several positions at the same time, multiple heads are used, which are then joined and projected into a smaller space to keep the computational cost similar to basic attention.

Taking head attention. In traditional multi-head attention the different attention heads perform separate computations, while the greatest change in "Talking-Heads Attention" is breaking that separation. In fact are inserted two further learned linear projections, Pw and Pl, which transform the weight matrices and the attention-logits after the weight matrices multiplication, *moving information across attention heads.*
The inclusion of these "talking-heads" projections can achieve better results when is done transfer-learning to language comprehension and question answering tasks, so it should also affect performances of FNC-1 task.
As can be seen from *Figure 8* below , the greatest performance gains are achieved by increasing the number of heads, but for the project it was only possible to test BERT with 12 talking heads, which results to a relatively small boost in performance, passing from 82,6% of accuracy with Multi-head to 83.6% with talking head on MNLI from GLUE dataset.



	heads	$d_k = d_v$	SQuAD1.1 (f1)	MNLI
Multi-head	12	64	88.51	82.6
Talking-heads	6	128	88.8	83.4
Talking-heads	12	64	89.2	83.6
Talking-heads	24	32	89.4	83.6
Talking-heads	48	16	89.5	83.4
Talking-heads	64	12	89.9	83.8
Talking-heads	96	8	89.3	83.6
Talking-heads	192	4	89.8	83.9
Talking-heads	384	2	**90.5**	83.9
Talking-heads	768	1	**90.5**	**84.2**

Figure 7: Comparison of performances considering different number of talking heads.

5 Transfer Learning Through Fine Tuning

In this first section I evaluate the ability of previously analyzed model to adapt to the FNC-1 task by applying transfer learning through fine tuning.
The architecture is voluntary kept simple, without including additional complex layers on top, so that is possible to better appreciate how the pre-trained model can adapt to the downstream task. It is simply added a dropout layer to reduce overfitting, a dense layer, with a number of neurons equals the number of classes, and a Softmax to get a probability distribution from the logits.

5.1 Data preprocessing

Each pair *"Headline"*, *"articleBody"* given as input is concatenated into a single string using two special tokens, obtaining: "[HEAD] + Headline + [BODY] + articleBody".
These strings, before being in input to the BERT-like model, are transformed into numeric tokens IDs and are rearranged in multiple Tensor with a preprocessing function, provided by Tensorflow Hub, which is already configured with a vocabulary and its associated text normalization logic and needs no further set-up.
Specifically, the preprocessing function truncates each input to a fixed size and returns a dictionary with *inputWordIds*, which maps each token to the corresponding index in the vocabulary. The fixed size might appear to be a limitation of the model, as long text is truncated, but as noted in section 5.2 it has no significant effect.

5.2 Network Training and Hyperparameters choice

All the networks were trained with the same hyperparameters, in order to make a fair comparison among all, choosing the one suggested by the literature or by experiments.

Optimizer. The optimizer used is the one proposed by "Decouple Weight Deacy Regularization" (Loshchilov et al, 2019), that has been found to be very effective. It is a variation of Adam, that does the regularization by weight decay, without using moments.

Learning rate. Regarding the learning rate (Devlin et al, 2019) suggest a learning rate to fine tune BERT between 5e-5, 4e-5, 3e-5, and 2e-5, which is lower than the learning rate that is usually used in the pretraining task, i.e. 1e-4.
So I decided to use an initial learning rate equal to 3e-5 with a linear warm-up phase over the 10% of training steps, where the learning rate is linearly increased from 0 to the target, then, after that point, learning rate decay starts.

Seq length	**FNC-score**	**F1-macro**	**F1-micro**	**Inference time**	**Speedup**
128	83.9	**60.3**	89.7	**73**	x1
512	**84.4**	60.1	**90.2**	475	x7

Table 4: Comparison of performances, considering different sequence length in input to the small-BERT model.

Seq length	**agree**	**disagree**	**discuss**	**unrelated**
128	**56.8**	**8.1**	78.2	98.1
512	55.4	7.9	**78.5**	**98.7**

Table 5: Comparison of F1 measure on each class, considering different sequence length in input to the small-BERT model.

Epochs and Batch size. (Devlin et al, 2019) suggest a batch size between 16 and 32 and a number of epoch among 2, 3, 4 to fine tune BERT.
Based on this suggestion I chose a batch size equal to 32, except for BERTlarge and BERTtalking where I used 8 as batch size for low memory issues due to lack of sufficient resources. While for the number of epochs I have chosen 3 to find a good compromise for the training time.

Sequence length. As mentioned in *section 5.1*, all Transformer encoder receives input after that they are transformed, with the API provided by Tensorflow, into a suitable form to be processed. But an important parameter to set is the sequence length of the text that the encoder has to receive in input, in fact the longer sequences are simply truncated, thus resulting into a partial view of longer article body.
The average number of tokens for the headline is 11, while for the article body it is 370, so I decided to try a sequence length equal to 512, so that the network could receive almost any input without truncation, and equal to 128, which is the default dimension provided by the API.
As can be seen in *Table 1* and *Table 2*, obtained with experiment using the validation set with the fine-tuned small-BERT model, there are no significant drop in performances by reducing the sequence length, in fact the FNC score and the F1-macro are reduced by only 0.2%, while the F1-micro only by 0.1%. Furthermore, it can be noted that using a sequence length of 128 also results in a slight increase of 0.2% in the F1 class on the *"disagree"* class, which is the most difficult due to lack of samples.
Finally I have decided to use sequence length equal to 128 for three main reasons, the first is that the performances are not affected, secondly increasing the sequence length greatly increases the inference time by a factor x4 and consequently also the training time is very high. The latter is a serious problem, since that I also have to test huge models, especially BERTlarge and BERTtalking,

with a limited amount of resources, thus resulting in a training time that is not affordable.
The last reason is practical as larger models with a longer sequence length could not be tested due to insufficient memory issues.

5.3 Result comparisons

As pointed out in *Section 3*, the most important metrics are FNC-score, F1-macro and F1 on each class, but for a complete view of perfomances, are also shown comparison about the F1-micro, the inference time and the speedup based on the fastest network.

Best models. Analysing *Table 3* and *Table 4*, we can see that, in gen-

MODEL	**Params**	**FNC-score**	**F1-macro**	**F1-micro**	**Inference time**	**Speedup**
BERT$_{base}$	108M	87.0	71.3	92.0	274	x4
BERT$_{large}$	334M	88.3	74.2	92.5	740	x10
BERT$_{expert}$	108M	**88.7**	**75.3**	**93.0**	242	x3
Small-BERT	29M	83.9	60.3	89.7	**73**	x1
BERT$_{talking}$	110M	87.6	73.0	92.4	319	x4
Albert	12M	87.1	72.0	92.1	241	x3
Electra	14M	85.1	61.5	90.2	92	x1

Table 6: Comparison of evaluation metrics with all the fine-tuned BERT-family models.

MODEL	**agree**	**disagree**	**discuss**	**unrelated**
BERT$_{base}$	65.0	38.5	83.0	98.6
BERT$_{large}$	**69.5**	45.4	83.2	98.9
BERT$_{expert}$	69.3	**48.6**	**84.4**	**99.0**
Small-BERT	56.8	8.0	78.2	98.1
BERT$_{talking}$	64.6	42.5	82.7	98.9
Albert	59.8	42.3	82.5	98.7
Electra	68.0	8.6	79.6	98.2

Table 7: Comparison of F1 measure on each class with all the fine-tuned BERT-family models.

eral, the larger the network the better the performance, in fact, as have also underlined (Devlin et al. 2019) during test on different BERT sizes against downstream tasks, increasing the number of parameters available to a network helps to learn better, even if the gradient of performance function according to size of the model is always smaller.

Despite this, the best model is not the largest, but is BERTexpert, so the one with a deeper preliminary knowledge on Language Inference task. In fact, as we could have guessed, the further fine-tuning carried out on a similar task and on a larger dataset has resulted in a base model that already knows how to capture the relationship between two pairs of text.
Considering the remaining models, it can be seen that the second best result is obtained with BERTlarge, but if we consider also other factors, such as number of parameters and speedup, it is not the global best model. Below are analyzed best models, considering trade-off between performances and speed/model size.

Trade-off model size and performances. Looking at the graph in *Figure 9*, we can focus on the relationship between the number of parameters and performance.
In particular, it can be noted that the best trade-off between number of parameters and FNC-score is obtained by Albert, in fact it has only 1.6% and 1.2% less respect to, rispectively, BERTexpert and BERTlarge, but with the difference that the latter has about 28 times more parameters.
Albert' remarkable results are likely due to the modification to the Next-sentence prediction task, that, as underlined in *Section 4.4*, has increased the network's ability to distinguish coherence between a pair of texts.
Very similar consideration can be done considering as metric the F1-macro, as shown in *Figure10.*

Trade-off speedup and performances. If we focus more on speed we can see that Albert is not so good, in fact it has inference time comparable to model with 9 times its number of parameters. But it is not surprising, since that the aim of Albert is not to make the model faster, but primarily to reduce the model size and increase performances.
Electra and Small-BERT are the ones that perform better considering a trade-off between score and time, in fact they have just 3.2% and 4.4% of FNC-score less than BERTlarge, but with a reduction by a factor of ten of the inference time. Again, we could have anticipated this behavior, since that also (Clark et al, 2020) for Electra and (Turc et al, 2019) for Small-BERT underlined that one of their point of strength is the speedup.

Performances on single classes. Without any doubt, the hardest class is the *"disagree"* and it is mainly due to the fact that are available only 840 samples, which corresponds to the 1.7% of the entire dataset.
As shown in *Table 4*, all the models have unsatisfactory result below the 50%, but the worst performances are given by Electra and Small-BERT, that reach only 8% of F1 on this class, so it is the main cost to pay to have lower size and greater velocity.
Also the *"agree"* samples cannot be efficiently discriminated, as all the model have around the 60%-70% of F1 score.
On the other hand, all models can reach almost perfect score for the *"unrelated"* class, which composes the 72.2% of the dataset, with an F1 around 98-99%, and

fairly good scores, around 82%, on the *“discuss”* class, which is the second most present, composing the 17% of the dataset.

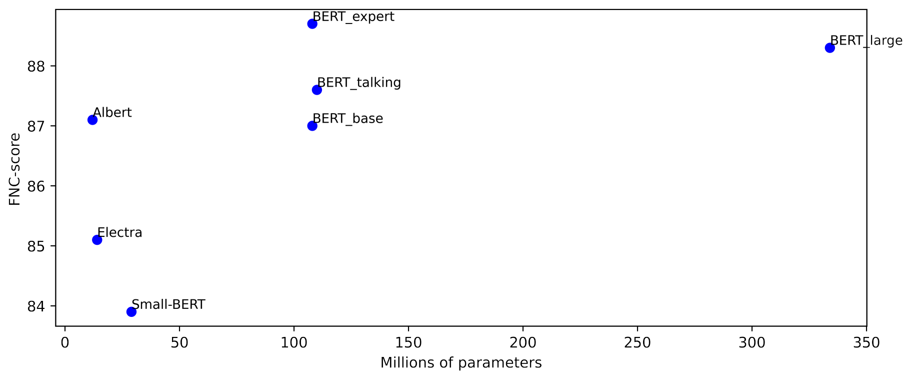


Figure 8: Scatter plot of FNC-score changing the number of parameters for all the model analysed.

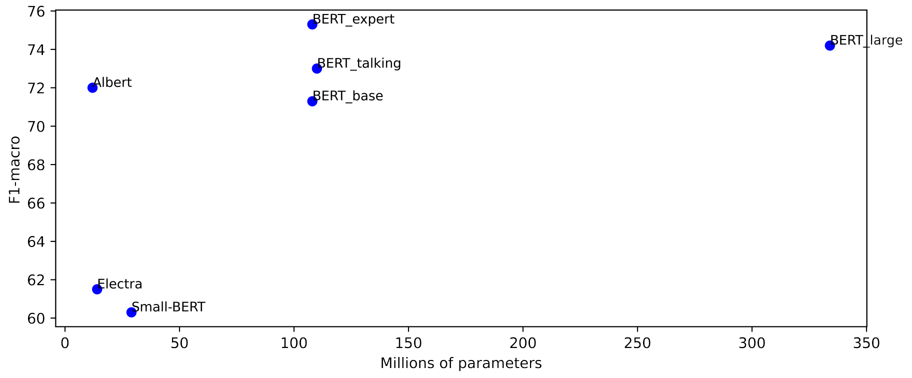


Figure 9: Scatter plot of F1-macro changing the number of parameters for all the model analysed.

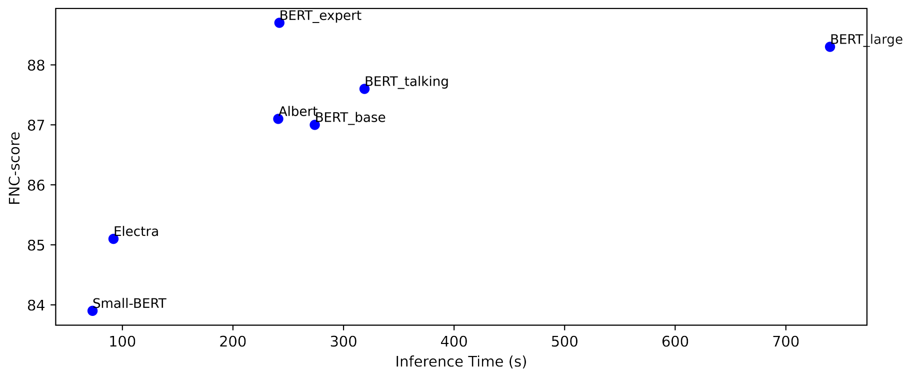


Figure 10: Scatter plot of FNC-score changing the inference time for all the model analysed.

6 Contextual Embeddings From Freezed Models

In this second section of the project I try to analyse the ability of the BERT-family models to produce contextual word embeddings, that are able to comprise the real meaning of words also considering the surrounding tokens.
These embeddings are given as input to a Convolutional Neural Network, whose architecture is inspired by one of the network that compose the ensemble model proposed by the “SOLAT in the SWEN” group, who won the FNC-1 challenge. It has been reimplemented, modified and improved considering slightly different configuration of the network and a different tuning of the hyperparameters.

6.1 Architecture of the Convolutional Neural network

Figure 13 shows the network architecture proposed by “SOLAT in the SWEN”, which receives as input the word embeddings of the headline and the articleBody and separately processes them with two identical convNet composed of 5 convolutional layers with an increasing number of kernels (256, 512 and 768), each followed by a dropout and a max-Pooling layer.
After being processed by the convNet, a Global Max-Pooling layer compresses the representation of the headline and the articleBody into two vectors of 768 features, that are then combined and given as input to a multi-layer perceptron composed by three dense layers with 1024 neurons and a final layer with 4 neurons, as the number of classes.
Finally, with a softmax layer, is obtained a probability distribution from the logits, which is used to determine the output class of the pair of texts in input. Note that all the activation functions are ReLU.

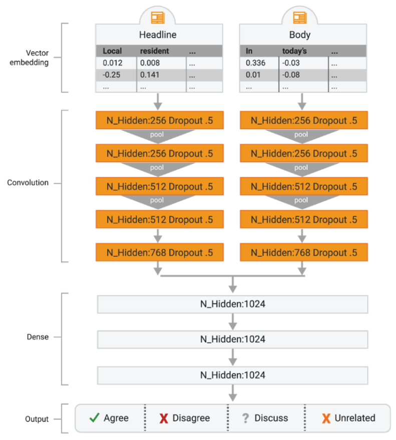


Figure 11: Architecture of the Convolutional network that will process the contextual embeddings produced by the BERT-family models.

6.2 Experiments on the convolutional network

To obtain baseline results and to ensure a fair comparison between all contextual embeddings, without optimizing the hyperparameter for a specific network, was used for the experiments word embeddings obtained from the python spacy library.
All tests were carried out using the *validation* set to compute the metrics score. The three most interesting ones are reported in *Table 5*, where can be seen that the proposed network was already able to get satisfactory results, about 87% on the FNC score and 66% on the F1 macro.

But, considering different values of window length, which was originally set to

Network config	**FNC-score**	**F1-macro**
base	87.2	66.4
window length=5	89.3	**70.8**
augmented capacity	**90.2**	67.6

Table 8: Comparison of the evaluation metrics with the three most significative tests on the CNN.

Network config	**agree**	**disagree**	**discuss**	**unrelated**
base	68.5	21.0	80.7	95.6
window length=5	**74.5**	**25.7**	**87.1**	**96.0**
augmented capacity	72.2	18.5	85.6	94.3

Table 9: Comparison of F1 measure on each class with the three most significative tests on the CNN.

three, I found a better value equal to 5, with which I was able to get 2.2% and 4.4% more on the FNC score and F1 macro respectively.
Another remarkable result was obtained considering an expansion of the network capacity, increasing the number of neurons in the dense layer to 2048, in fact it led to 0.9% more on FNC score. However, considering that all the performances of F1 on single classes decrease, and consequently the F1 macro, I decided that this configuration was worse than the one without the enlargement.

6.3 Network training

Also in this case, all the architectures were trained with exactly the same conditions, in order to have fair comparisons.

Embeddings extraction. To extract word embeddings I followed the approach described in (Devlin et al, 2019), where BERT models are frozen, making all parameters untrainable, and are used only for feature extraction. The latter can be extracted in different way from the model, but I decided to *concatenate the last four hidden layers*, as it was the most effective methods found in the paper.

Hyperparameters. All the models were trained for at most 20 epochs, unless the validation loss did not improve for three consecutive epochs, in fact in that case early stopping interrupts the training. As optimizer was used Adam, with a learning rate equal to 2e-4.

6.4 Result comparisons

As we can see in *Table 10* there are surprising results, as many FNC-score obtained using contextual word embeddings from BERT-family models seems to perform almost like Spacy word embeddings.

Best models. BERTbase is undoubtedly the winner, as it can beat all the others both on FNC-score and F1-macro reaching respectively 65.7% and 42.6%. Another model that was able to get good performances is BERTtalking, as it could reach 40.9% on F1-macro and is also the best in classifying the ”disagree” class, which has given much more problem in this second type of experiment, as many models cannot classify its instances at all.

The fall of the King. In this case the BERTexpert model was not able to extract good contextual word embeddings, as it can reach 60.2% on FNC-score and 39.1% on F1-macro, that is, respectively 5.5% and 3.5% less than BERTbase, that has exactly the same architecture and number of parameters. So, it seems that the further fine tuning on the MNLI task does not help at all, on the contrary, it worsens performances.

Trade-off model size/speedup and performances. Looking at the figure

MODEL	Params	FNC-score	F1-macro	F1-micro	Inference time	Speedup
Pretrained embeddings	-	60.0	34.8	53.5	11	-
BERT$_{base}$	108M	**65.7**	**42.6**	70.1	569	x3
BERT$_{large}$	334M	61.8	40.3	67.6	1704	x8
BERT$_{expert}$	108M	60.2	39.1	66.4	563	x3
Small-BERT	29M	60.4	39.9	69.0	**202**	**x1**
BERT$_{talking}$	110M	62.5	40.9	68.8	743	x4
Albert	12M	39.4	21.0	**72.2**	623	x3
Electra	14M	57.4	37.7	64.8	203	**x1**

Table 10: F1 on each class performances for different convNet configuration on FNC-1 task.

MODEL	**agree**	**disagree**	**discuss**	**unrelated**
Pretrained emb.	26.9	0.0	44.8	67.6
BERT$_{base}$	**35.5**	0.0	**51.6**	83.3
BERT$_{large}$	30.1	0.2	48.1	82.8
BERT$_{expert}$	29.3	0.7	46.6	79.9
Small-BERT	27.4	0.0	50.2	82.1
BERT$_{talking}$	30.4	**1.4**	48.5	83.3
Albert	0.0	0.0	0.1	**83.9**
Electra	27.3	1.1	44.6	77.9

Table 11: F1 on each class performances for different convNet configuration on FNC-1 task.

12, 13 and 14 it can be noted that considering the number of parameters and the inference time, the best results are obtained by Small-BERT, which manages to get only 5.4% and 2.7% less on the FNC and F1-macro score than BERTbase, but with a reduction of the number of parameters by a factor of 4 and with three times less inference time.

Analysis on FNC-score. The critics made by (Hanselowsky et,. al) on this metric are now much more evident, in fact at a first view it may seems that 4 contextual word embeddings on 7 perform almost like or worse than pre-trained word embeddings. But analysing the results further, we can conclude that only Albert model is really worse, in fact it can be seen that spacy embeddings can reach only 34.8% on F1-macro score and gives always worse results considering F1 on single classes.
The high FNC-score of spacy embeddings is due to the fact that the network is much better to classify *"related"/"unrelated"*, rather than discrimining the right class, and, as pointed in *Section 3*, it is a big limit, in fact it also never try to classify an instance as *"disagree"*. Looking at the confusion matrix in *Figure 14*, can be noticed that most of classes are correctly classified as related or *"unrelated"*, thus obtaining a score of 0.25, but then the model cannot chose the right related class.
While considering the *Table 11* can be noticed that Electra, even if reached 2.6% less on FNC score, can better classify the single classes, even the *"disagree"* one.

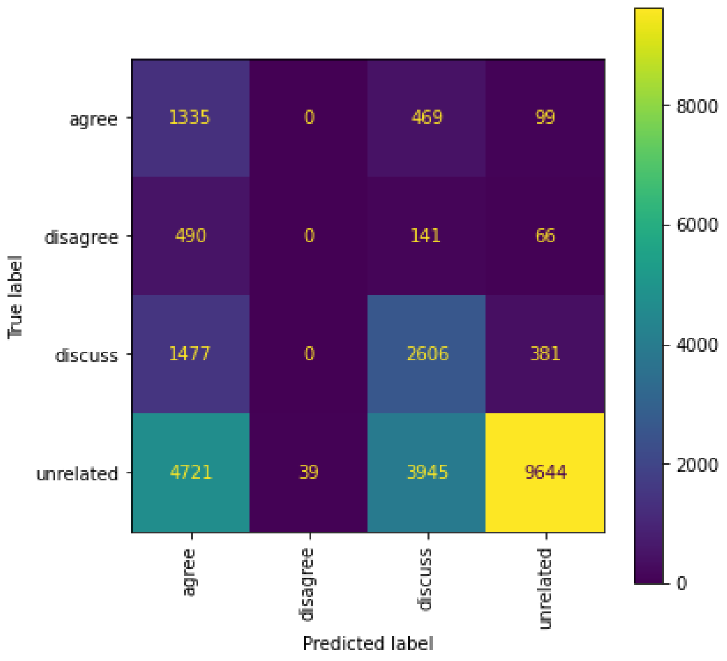


Figure 12: Confusion matrix produced by predictions with Spacy embeddings.

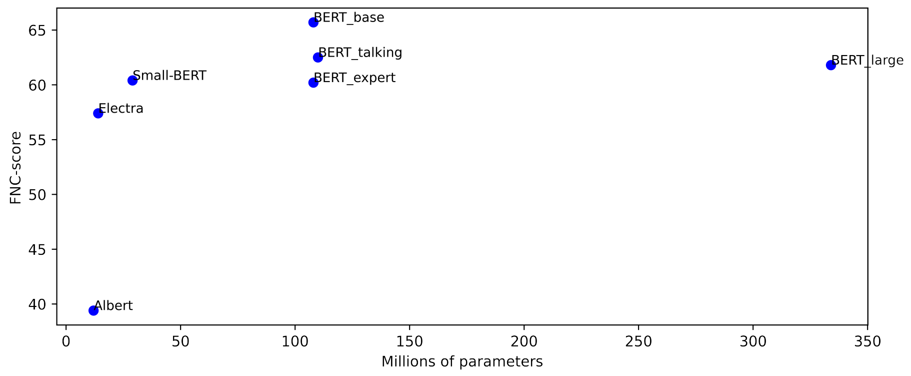


Figure 13: Scatter plot of FNC-score changing the number of parameters for all the model analysed.

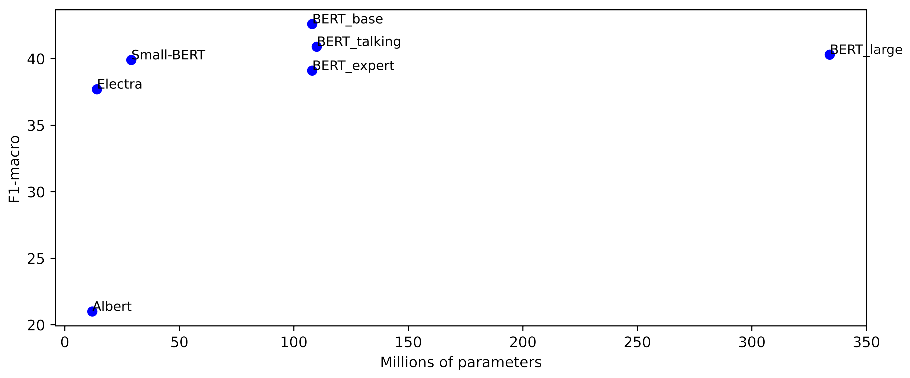


Figure 14: Scatter plot of F1-macro changing the number of parameters for all the model analysed.

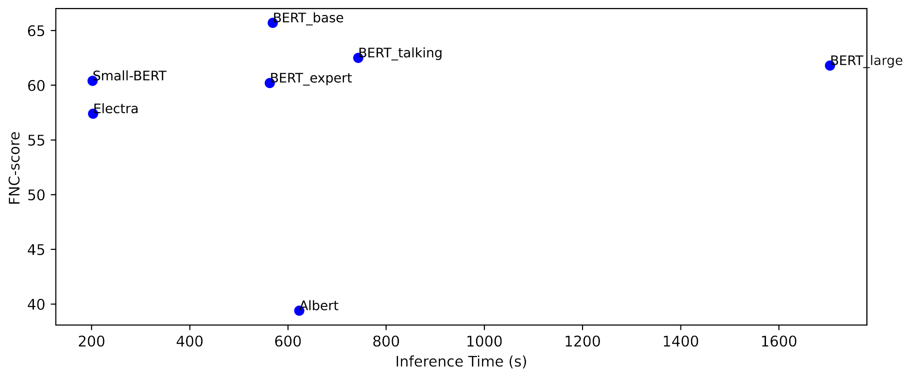


Figure 15: Scatter plot of FNC-score changing the inference time for all the model analysed.

7 Conclusions

The analysis on the performances of fine-tuned pre-trained BERT-family model has again confirmed their incredible ability to generalize well on task different respect to the one on which are pre-trained, also having a small dataset and after a low number of epochs of fine tuning.
We have seen that in general larger models perform better, even if, this time, the "intelligence" has beat the "force", as BERTexpert model was able to give better performances respect to BERTlarge, even with almost 3 times less parameters.

Considering the quality of the contextual word embeddings extracted from these models, the analysis showed that they can lead to better results than pre-trained word embeddings, and that good performance in fine tuning is not always reflected in good performance in features extraction, in fact the BERTbase model manages to achieve better results than BERTexpert and BERTlarge. Perhaps this difference is due to the way in which contextual word embeddings are extracted, as the experiments done by (Devlin et al, 2019) concerned the BERT-base model.

References

[1] Hanselowski, A., PVS, A., Schiller, B., Caspelherr, F., Chaudhuri, D., Meyer, C. M., and Gurevych, I. (2018). Repository of the COLING 2018 paper: A Retrospective Analysis of the Fake News Challenge Stance Detection Task. *https://arxiv.org/pdf/1806.05180.pdf*

[2] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805v2 [cs.CL] 24 May 2019.*

[3] Valeriya Slovikovskaya, Giuseppe Attardi. Transfer Learning from Transformers to Fake News Challenge Stance Detection (FNC-1) Task. *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 1211–1218.*

[4] Iulia Turc, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. Well-read students learn better: on the importance of pre-training compact models. *arXiv:1908.08962v2 [cs.CL] 25 Sep 2019.*

[5] Zhenzhong Lan Mingda Chen2 Sebastian Goodman1 Kevin Gimpel2 Piyush Sharma1 Radu Soricut1. Albert: a lite BERT for self-supervised learning of language representations. *arXiv:1909.11942v6 [cs.CL] 9 Feb 2020.*

[6] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531, 2015.*

[7] Yunjiang Jiang, Yue Shang, Ziyang Liu, Hongwei Shen, Yun Xiao, Wei Xiong, Sulong Xu, Weipeng Yan, Di Jin. BERT2DNN: BERT Distillation with Massive Unlabeled Data for Online E-Commerce Search.*arXiv:2010.10442v1 [cs.LG] 20 Oct 2020.*

[8] Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning. Electra: pre-training text encoders as discriminators rather than generators. *Published as a conference paper at ICLR 2020.*

[9] Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. Language GANs falling short.*arXiv preprint arXiv:1811.02549, 2018.*

[10] Noam Shazeer. GLU Variants Improve Transformer. *http://arxiv.org/abs/2002.05202v1.*

[11] Noam Shazeer, Zhenzhong Lan, Youlong Cheng, Nan Ding, Le Hou. Talking-Heads Attention. *arXiv:2003.02436v1 [cs.LG] 5 Mar 2020.*

[12] Ilya Loshchilov Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101v3 [cs.LG] 4 Jan 2019.*