Evaluating Unsupervised Representation Learning for Detecting Stances of Fake News

Maike Guderlei
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
maike@guderlei.de

Matthias Aßenmacher
Department of Statistics
Ludwig-Maximilians-Universität
Munich, Germany
matthias@stat.uni-muenchen.de

Abstract

Our goal is to evaluate the usefulness of unsupervised representation learning techniques for detecting stances of Fake News. Therefore we examine several pretrained language models with respect to their performance on two Fake News related data sets, both consisting of instances with a headline, an associated news article and the stance of the article towards the respective headline. Specifically, the aim is to understand how much hyperparameter tuning is necessary when finetuning the pretrained architectures, how well transfer learning works in this specific case of stance detection and how sensitive the models are to changes in hyperparameters such as batch size, learning rate (schedule), sequence length as well as the freezing technique. The results indicate that the computationally more expensive autoregression approach of XLNet (Yang et al., 2019) is outperformed by BERT-based models, notably by RoBERTa (Liu et al., 2019). While the learning rate seems to be the most important hyperparameter, experiments with different freezing techniques indicate that all evaluated architectures had already learned powerful language representations that pose a good starting point for finetuning them.

1 Introduction

With the rise of social media, exchange of opinions and news happens faster than ever. News circulation is therefore less and less bound to traditional print journalism that usually requires extensive research, fact checking and accurate coverage in order to be a reliable news resource. It is relatively easy to share opinions that are either not supported by researched facts or simply wrong. In the worst case a large amount of people can be targeted by propaganda in order to shift societal discussions in favor of a wanted agenda. Human resources are limited to identify such Fake News, since they cover a wide range of topics and linguistic writing styles (Shu et al., 2017, p.2). Automated Fake News detection (FND) has therefore proven to be an important challenge for NLP researchers in recent years. To this day, a variety of approaches dealing with FND exists (Khan et al., 2019).

In 2017, the *Fake News Challenge Stage 1* was introduced which tackles FND as a stance detection task (Pomerleau and Rao, 2017), where the idea is to determine the stance of a news article to a given headline (Hanselowski et al., 2018, p.1). We now evaluate the five models BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), ALBERT (Lan et al., 2019) and XLNet (Yang et al., 2019) which were developed and enhanced recently. All architectures were pretrained on large unlabeled corpora using a self-supervised objective and can be finetuned on a desired task at hand. Our main focus is to evaluate the necessity of hyperparameter tuning as well as the general performance.[1] Besides this, it is of special interest to examine the differences between auto-encoding (BERT-based models) and auto-regressive architectures (XLNet).

In this context, the term Fake News is defined as a text piece that is verifiably wrong and spread with a malicious intention. In doing so, other media sources such as video, images or audio are excluded. Since the intention has to be malicious all sorts of entertainment related false news such as hoaxes and april fools are excluded. The definition is similar to the narrow definition that Shu et al. (2017) undertake.

[1] Code and data sets available on GitHub: https://github.com/magud/fake-news-detection

Proceedings of the 28th International Conference on Computational Linguistics, pages 6339–6349
Barcelona, Spain (Online), December 8-13, 2020

2 The Fake News Challenge (FNC-1)

In 2017, the *Fake News Challenge Stage 1* (FNC-1) was published. Organizers from industry and academia created an online challenge accessible via http://www.fakenewschallenge.org/ (Pomerleau and Rao, 2017). The FNC-1 is conceptualized as an important pre-step in identifying Fake News and exploring how artificial intelligence tools can be leveraged in combatting them. Given a certain claim about a topic, what are different news agencies reporting about this claim? If most news agencies agree with a claim, this can be interpreted as an indicator of the truthfulness of the claim. On the contrary, if a lot of news disagree with the claim, the claim is likely Fake News. Statistically speaking this idea is translated into a stance detection task with the claim being treated as a headline and the stance of the article body being either *Agree*, *Disagree*, *Discuss* or *Unrelated*. FND is thus treated as a classification task with four categories which are interpreted as the stance of an article body towards a given headline claim. Along with a baseline model, a training and test set was published. According to Hanselowski et al. (2018), 50 teams participated in the challenge.

3 Related Work

In general, FND can be interpreted as a binary classification task where a text document is classified as either *Fake News* or *No Fake News* or as a multi-class problem often with ordinal labels. One major distinction in different classification algorithms can be drawn in the consideration of auxiliary information. Most contributions take a purely feature-oriented approach, while others try to incorporate information revolving around spreading Fake News and the respective dissemination process. The feature-based approach focuses on extracting relevant linguistic peculiarities associated with Fake News. Typical examples are characters n-grams, words or a measure of the readability and syntax of an article body (Pérez-Rosas et al., 2018, p.5). Other authors included the average word length, counts of exclamation marks or the sentiment of an article (Khan et al., 2019, p.6). After finding appropriate features, traditional machine learning algorithms as well as (deep) neural networks are proposed to classify the text instance given the extracted features. Other feature-based approaches directly rely on deep learning, using neural networks for learning good representations of the text input by a stack of hidden layers which is then fed in a last classification layer. This approach is used by Yang et al. (2018) who simultaneously train a CNN on text *and* image data to classify fake entities and by Dong et al. (2019) who implement a two-step approach of using supervised and unsupervised learning with a CNN as well. But despite the fact that feature-based approaches are fairly popular within the FND research, Shu et al. (2017) argue that they are not sufficient. Approaches that use auxiliary information e.g. revolve around modeling the dissemination process of Fake News by incorporating spatio-temporal information about users who like, share or publish (potential) Fake News (Ruchansky et al., 2017; Ren and Zhang, 2020). The FNC-1 organizers use a feature-based approach by proposing a baseline model that extracts various features from the headlines and article bodies. The approach of specifically modeling the relationship between a headline and a respective article body was also exploited by Yoon et al. (2019) in the context of clickbait detection while the FNC-1 takes this approach as a pre-step to FND.

4 Material and Methods

4.1 Model architectures

Representation learning can be seen as one of the crucial success factors of large pretrained models such as BERT or XLNet that yield outstanding performances on a variety of NLP tasks. With the introduction of BERT (Devlin et al., 2019) - short for Bidirectional Encoder Representations from Transformers - the possibility of *simultaneously* learning left and right word context was introduced. Up to this point, considering bidirectionality was only possible by modeling two separate networks for each direction that would later be combined (Peters et al., 2018). After learning deep bidirectional representations from unlabeled text, BERT can be used for either finetuning or feature extraction. The model is pretrained on the combined objective of masked language modeling (MLM) and next-sentence prediction. The main premise of RoBERTa (Liu et al., 2019) is the assumption that BERT was seriously undertrained during

pretraining. RoBERTa is thus trained on larger batches of longer sequences from a larger per-training corpus for a longer time. In addition, Liu et al. (2019) argue that the second task of the next-sentence prediction does not improve BERT's performance in a way worth mentioning and therefore remove the task from the training objective. While RoBERTa focuses on improving pretraining, DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019) were introduced as lighter versions of BERT applying parameter reduction techniques in order to be usable under constrained computational training and inference budgets. Sanh et al. (2019) criticize the idea of simply enlarging data sets and running more and more exhaustive pretraining since this does not consider computational costs, memory requirements and even environmental aspects that are often neglected for the sake of further enhancing performance (Strubell et al., 2019; Hao, 2019; Peng, 2019). DistilBERT ends up diminishing BERT's architecture by around 40% while retaining around 97% of its language understanding capabilities and being 60% faster by using a distinct knowledge distillation approach (Hinton et al., 2015), while ALBERT relies on factorized embedding parameterization and cross-layer parameter sharing. Lan et al. (2019) specifically point out that simply enlarging model architectures by using bigger hidden size dimensions or more layers can lead to unexpected model degradation[2]. While RoBERTa, DistilBERT and ALBERT focus on either improving or reducing the size of BERT, the overall bidirectional encoder architecture with the MLM objective remains the same. With XLNet, Yang et al. (2019) propose an alternative approach that works in an auto-regressive manner. Since BERT's objective is to reconstruct a corrupted input, it can be described as a denoising autoencoder (DAE) approach. In contrast to the denoising autoencoder, auto-regressive (AR) language modeling uses a sequential token prediction that can only condition on either left or right context. XLNet combines the advantages of both approaches, namely the bidirectionality while capturing dependency structures among tokens better by employing a so called permutation language modeling objective (PLM).

The pretrained models are all implemented in PyTorch (Paszke et al., 2019) using the *huggingface* `transformers` library (Wolf et al., 2019) that makes a large variety of the SOTA models in NLP available and ready to use. Architecture-wise, the `base-cased` implementations with the suitable head for sequence classification (i.e. `<model_name>ForSequenceClassification`) are used. For DistilBERT and XLNet the weights of the pooling layers are randomly initialized, while BERT, RoBERTa and ALBERT rely on pretrained pooler layers. The weights for the softmax classification layer are randomly initialized for all five models. For DistilBERT a model version that is distilled from the RoBERTa base model was chosen. All models are finetuned using an Ubuntu 18.04.3 LTS OS image with 40 CPUs (Intel Xeon, 2.4 GHz clock speed), 736 GB of RAM and a maximum of two Tesla V100-PCIE-16GB GPUs.

All models were finetuned using the Adam algorithm with default values for its hyperparameters as indicated by Kingma and Ba (2014). As warmup ratio, we chose a value of 0.06 complemented by a more extensive evaluation when it comes to the learning rate schedule, where we conduct small grid search experiments later on. In order to avoid gradient explosion, the norm of all gradients is clipped with a maximum value of 1. Gradients are not accumulated. For all models, the pretrained weights of the specified *huggingface* version are loaded once. The experiments and grid search steps thus use the same randomly initialized weights for the finetuning layers per model[3].

4.2 Data

The FNC-1 data set was created from the Emergent data set (Ferreira and Vlachos, 2016) which was developed for an online journalism project about rumour debunking. The project is still running and a website with manually checked claims is available (Silverman, 2019). Rumours were extracted from websites such as *snopes.com* and twitter accounts such as *@Hoaxalizer*. Journalists then first identified the respective claim and searched for articles mentioning this claim. As a next step the journalists labeled

[2]However, the performance of BERT, RoBERTa and XLNet can only be exceeded when the model is upscaling its width again which is contrary to the authors initial idea of presenting a leaner model architecture.

[3]Please note that the term *finetuning layer* refers to the additional layers that are put on top of the main model architecture, while *pretraining layers* are those layers that are part of all models no matter which finetuning task is used. Which layers were updated during the finetuning is not indicated by these terms, but is rather defined by the specified freezing technique.

the article as *For*, *Against* or *Observing* and then summarized the article into a headline. As an additional step the veracity level of the claim was labeled as *True*, *False* or *Unverified*. In total, 300 rumoured claims and 2,595 associated news with an average ratio of 8.65 (sd = 7.31) articles per claim were considered. The data set thus contains real world data which was manually labeled by journalists with regard to their stance and veracity level. For the FNC-1, organizers matched every article body with its respective headline and additionally created the fourth class *Unrelated* by randomly matching headlines and article bodies that belonged to different topics. Furthermore, 266 instances were created in addition to prevent participation teams from deriving labels for the test set since the Emergent data set is publicly available. The class distribution over the four classes in the FNC-1 data set is heavily skewed towards the *Unrelated* class. The 49,972 instances each consist of a headline (i.e. the claim to be looked at), the respective article body and label. In total, there are 1,669 unique article bodies and 1,648 unique headlines. The 300 topics are divided into 200 topics for training and 100 topics for testing. Not every claim is associated with all four labels.

Hanselowski et al. (2018) introduce the extended data set FNC-1 ARC as follows: ARC consists of 188 manually selected debate topics of popular questions from the user debate section of the New York Times. For each of these debate topics those user posts were selected that were highly ranked by other users. These highly ranked user posts were processed by producing two opposing claims for them. Afterwards, crowd workers decided on the stance of the user posts with regard to the two opposing claims and labeled the post as either *Agree*, *Disagree* or *Discuss*. The *Unrelated* label was created by randomly matching user posts to different topics. As this data set is built on user posts as opposed to the online news articles of the original FNC-1 data set, it consists of shorter documents that tend to express one viewpoint only and are less balanced in their opinion as news articles. Using this additional data set, the robustness of the provided models can thus be tested. The extended FNC-1 ARC data set combines the FNC-1 with the ARC data set. It comprises 64,205 instances, 14,233 more than the FNC-1 data set. The label distribution is overall similar to the original data but the *Disagree* category has a bigger proportion.

4.3 Pre-Processing

As a first step, headlines and article bodies are concatenated into one long sequence with headline coming first and the respective article body following. In doing so, the models can be evaluated with respect to their capabilities of learning the semantic structures of one instance as a whole. All model architectures can be used with their own respective tokenizer. Since every tokenizer provides the possibility to process cased text, it was decided to not perform lower-casing. All tokenizers can detect and ignore control characters. It was thus not necessary to explicitly remove them. The removal of seemingly uninformative words, so called stop words, was kept as low as possible and consisted of a manually selected list containing the words *The*, *the*, *A*, *a*, *An*, *an*. In order to remove stop words, it was necessary to first tokenize the instances to be able to filter out the mentioned stop words. After tokenizing each instance, the remaining tokens were padded. Instances that have a longer sequence length than 512 were truncated. As a final result, all tokens have a sequence length of $T = 512$ including all necessary special tokens. In addition, the model is fed with an identifier of which token is padded and which is not. This is necessary to only place the attention over the span of ”true” tokens that is of non-padded tokens.

5 Results

5.1 Initial Experiments

The goal of the exploration step is to evaluate the general performance and gain insights as to how useful the given recommendations for hyperparameter choices are. Therefore, the main hyperparameters of interest, namely the sequence length, batch size, learning rate and learning rate schedule are kept fixed to determine how well the models perform with respect to different freezing techniques. The drawn conclusions are then considered for the grid search. The exploration step takes the full maximum sequence length that is available for each model which consists of 512 tokens. The common assumption for batch sizes is that a higher batch size yields a more accurate estimate of the gradients (Goodfellow et al., 2016, p.276). For the given hardware resources this results in a batch size of 8 which is the best

possible value for the given sequence length and memory capacity of the hardware. The learning rate is set to 3e-5 with a linear schedule. The number of epochs is 2 since it is not the goal of the experimental step to receive the best possible performance but rather to gain first insights into the general adaptability of transfer learning for the stance detection of Fake News.

Concerning freezing, different degrees can be considered: It is obviously possible to finetune all layers, that is to use the pretrained weights as starting point and then update all parameter weights during finetuning. The other extreme would be to only train the additional task-specific layers that are placed on top of the general model structure. For the setup of the initial experiments, three different versions are considered: The first one we call *Freeze*, meaning that all layers except for the last projection as well as the final classification layer are frozen. Second, *No Freeze* uses no freezing at all which means that all parameters are updated during finetuning, while the last approach *Freeze Embed* is done with frozen embedding layers only. For every freezing technique finetuning was performed three times. The results are given in Tab. 1 and reported with respect to the mean macro-averaged F_1-m metric which is known to handle imbalanced class distributions more effectively compared to the accuracy.

	FNC-1			FNC-1 ARC		
	Freeze	No Freeze	Freeze Embed	Freeze	No Freeze	Freeze Embed
BERT	20.88	**75.62**	74.93	21.06	75.16	**75.31**
RoBERTa	20.88	79.27	**81.72**	21.00	**78.89**	77.64
DistilBERT	20.88	**76.57**	76.46	21.00	75.31	**76.62**
ALBERT	34.66	67.91	**68.16**	34.70	69.49	**69.97**
XLNet	27.51	80.95	**82.18**	25.72	**80.20**	78.88

Table 1: Mean F_1-m metric over three runs for the exploration step. For *Freeze* only the last projection and classification layers are updated. For *No Freeze* all layers are updated, while for *Freeze Embed* all embedding-specific layers are excluded from updating. Results are on the dev set of a train/dev split of the actual training set. An overview on the runtimes can be found in Tab. 5 in Appendix A.

The results indicate the importance of not freezing too many layers. All models have problems to accurately learn when the main layers (id est the encoder layers for the BERT-based and the relative-attention with feed-forward layers for XLNet) are frozen. Most models still predict every instance as *Unrelated* after two epochs. ALBERT performs best in this setting. This is not surprising since all encoder layers in ALBERT share weights and the model thus learns less information in general. Freezing the encoder layers therefore leads to less information loss for ALBERT. XLNet performs slightly better compared to BERT, RoBERTa and DistilBERT. For both *No Freeze* and *Freeze Embed* all models are able to accurately learn to classify the given data sets. For the FNC-1 data set, the performances of XLNet and RoBERTa are very close. Overall, there is a slight tendency for models to perform better when only the embedding layers are frozen. The main takeaway is that only finetuning the task-specific layers is not sufficient. Between not freezing any layers and freezing the embedding related layers, the difference in performance with respect to the F_1-m metric is often neglectable. For the grid search, all models are finetuned with frozen embedding layers since this bears the advantage of speeding up training. Furthermore, models are trained for 3 instead of 2 epochs for the grid search.

5.2 Detailed Grid Search

Since traditional research on hyperparameter optimization focuses on training a model from scratch, some of the generated insights and algorithms might not be appropriate for the setting of transfer learning and finetuning (Li et al., 2020). There is little research on the effectiveness and necessity of hyperparameter optimization when finetuning, especially for NLP tasks. The biggest difference between pretraining and finetuning lies in the initialization of the weights. In the case of finetuning this initialization relies on the pretrained model which hopefully captures some intrinsic knowledge, whilst the pretraining phase usually works with random initialization. The goal of this experimental setup is thus to

gain more insights on the effectiveness and necessity of hyperparameter tuning for finetuning in an NLP context. Therefore, a grid search for the hyperparameters batch size, maximal sequence length and type of learning rate schedule and learning rate is conducted as indicated in Tab. 2.

The learning rate is the most common hyperparameter that is tuned. It is so important that Goodfellow et al. (2016) state to only tune the learning rate, if one has only time/budget to address one hyperparameter. For the finetuning setting, it is usually assumed that a drastically smaller learning rate should be used in comparison to pretraining (Li et al., 2020, p.1). During pretraining the model already learns fairly good representations which are valuable for any downstream task. These should therefore not be destroyed by using a too big learning rate that strides away too fast from the already gained knowledge. Thus, learning rate values of 1e-5, 2e-5, 3e-5 and 4e-5 are considered.

Hyperparameter	**Considered Configurations**
Batch size/Sequence length	16 / 256; 32 / 256; 4 / 512; 8 / 512
Learning rate	1e-05; 2e-05; 3e-05; 4e-05
Learning rate schedule	constant (*cst*), linear (*lin*), cosine (*cos*)

Table 2: Search space over chosen hyperparameters. The sequence length and batch size depend on one another due to memory capacity reasons. For the longer sequence length only smaller batch sizes could be considered. All learning rate schedules use a warmup period of 6% of the total optimization steps.

In addition, the schedule that is used along with the learning rate itself plays an important role. In this context, three different schedule types are considered, namely a constant, linear and a cosine schedule. All three types of schedules use the same warmup strategy for which a lower learning rate is used at the start of training to overcome optimization difficulties. After a warmup period the targeted learning rate is reached. The schedules now differ in the successive handling of the learning rate. The constant schedule keeps it at the targeted value, while the linear and cosine schedule decay it accordingly.

For the sequence length the naive assumption is that using as much context as possible is beneficial for the performance. A longer sequence length that does not truncate input sequences might therefore perform better. On the other hand it is imaginable that news articles might not need to be fed fully to the model since they might contain redundant discussion parts. Often, the main arguments are already shared at the beginning of the article with the full article further elaborating on the initially made statements. It might be sufficient to look at the beginning of articles which translates to a shorter sequence length. For the grid search, sequence lengths of $T_1 = 256$ and $T_2 = 512$ are examined.

Lastly, a shorter sequence length bears the possibility to increase the batch size which was found to be rather low in the exploration step due to limited memory capacities. A larger batch size is usually affiliated with a more accurate estimate of the gradient (Goodfellow et al., 2016, p.276). On the contrary, smaller batch sizes often have a regularizing effect which might be due to the additional noise they add to the learning process. The values for the batch sizes are chosen such that a constant tokens per batch ratio is reached. Given the memory constraints, the highest possible batch sizes are 32 and 8 for a a sequence length of $T_1 = 256$ and $T_2 = 512$ respectively. Given the defined search space in Tab. 2, 48 combinations are evaluated for each model and data set. The combinations are examined per learning rate and presented in Tab. 3.

When it comes to the sequence length, a value of 256 is preferred by most models for the FNC-1 data set, while for the FNC-1 ARC data set there is no preferred choice. This means that even though, the FNC-1 data set contains longer sequences on average, a shorter sequence length is often preferred. This can be interpreted as an indication that the similarity of instances is more important than the average sequence length. Apart from the fact that the FNC-1 ARC data set contains more instances compared to the FNC-1 data set, the biggest distinction is that the additional instances were generated from a different context. Hanselowski et al. (2018) specifically introduced the extended data set to test a proposed model's robustness by using the more heterogeneous FNC-1 ARC data set. The general performance on the different data sets is elaborated at a later point when discussing the indications of Tab. 4.

For a sequence length of T_2 = 512 a batch size of 8 leads to the best performance more often. The only exceptions are found for a smaller learning rate of 1e-5 or 2e-5. This is not surprising since a smaller batch size introduces more uncertainty in estimating the gradient. This uncertainty is then compensated by a smaller learning rate. For a sequence length of T_1 = 256 the affiliated preferred batch size is more evenly distributed. For the FNC-1 data set a sequence length of 256 is chosen 14 times in total with eight configurations preferring a batch size of 16 and six configurations one of 32. Hence, for the FNC-1 data set a smaller batch size is preferred for a sequence length of 256. For the FNC-1 ARC and the same sequence length a similar even distribution can be reported, with the batch size 16 being chosen four and the batch size 32 being chosen five times. There is thus an indication that the batch size seems to be especially important for longer sequence lengths where a higher batch size is preferred unless the models are trained on a very small learning rate. For both data sets a higher batch size of 32 tends to occur along with a higher learning rate of 4e-5.

		BERT		RoBERTa		DistilBERT		ALBERT		XLNet	
	LR	Winner	F_1-m	Winner	F_1-m	Winner	F_1-m	Winner	F_1-m	Winner	F_1-m
FNC-1	1e-5	16,256,cos	62.46	4,512,lin	**78.18**	8,512,cst	65.72	4,512,lin	56.62	4,512,cos	73.47
	2e-5	16,256,cst	**70.18**	16,256,lin	76.54	16,256,lin	67.64	8,512,cos	59.74	16,256,cos	**75.00**
	3e-5	16,256,cst	69.36	32,256,cos	76.52	32,256,cst	69.64	16,256,lin	**59.80**	32,256,cos	73.27
	4e-5	8,512,lin	68.09	32,256,lin	74.84	32,256,cst	**72.11**	16,256,lin	58.33	32,256,lin	73.46
FNC-1 ARC	1e-5	8,512,lin	68.87	4,512,lin	**78.19**	8,512,lin	71.99	8,512,cst	63.40	4,512,lin	74.42
	2e-5	4,512,lin	**72.20**	8,512,lin	77.27	8,512,cst	**73.59**	8,512,cos	**65.01**	8,512,lin	75.47
	3e-5	8,512,cos	70.93	16,256,lin	77.54	32,256,lin	72.99	16,256,lin	64.67	16,256,lin	73.97
	4e-5	32,256,lin	70.83	32,256,lin	77.54	16,256,lin	73.13	32,256,lin	63.63	32,256,lin	**75.57**

Table 3: Overview over the results of the grid search with respect to the learning rate (LR) in the left. *Winner* denotes the winning configuration (chosen with respect to F_1-m on the evaluation set) out of the 12 possible configurations per LR. The values in the *Winner* column indicate batch size, sequence length and LR schedule (abbreviated by *cst* for constant, *lin* for linear and *cos* for cosine) in this order. The F_1-m of the winning configuration per model is indicated in bold, values in teal indicate the overall winning configuration per data set. The overall winning configuration over both data sets (RoBERTa; F_1-m = 78.19) is additionally marked by a box. All reported values are obtained on the official test set.

Evaluating the learning rate schedule, the most surprising finding is that the linear schedule is chosen most frequently with being the preferred choice in 25 of all 40 reported winning configurations. The constant and the cosine schedule are very much on par with being chosen seven and nine times respectively. For the FNC-1 data set the distribution of the schedulers is more equal with nine configurations choosing the linear, five the constant and six the cosine schedule. For the FNC-1 ARC data set a clear preference towards the linear scheduler can be observed. It seems that a linear decay is sufficient but important for both data sets. There are no peculiarities when it comes to ALBERT which uses a different optimizer during pretraining.

When it comes to the learning rate, most models perform best for a smaller value of either 1e-5 or 2e-5. There is a difference in the two data sets however. For FNC-1 ARC the overall winning configuration is always either one of the two smaller learning rates for all models. Looking at the FNC-1 data set, DistilBERT and ALBERT yield an overall winning configuration with a learning of 4e-5 and 3e-5 respectively. For DistilBERT this can be explained by the fact that it only uses half the layers compared to its teacher RoBERTa which might require the model to stride away more from its pretrained version in comparison to the other models in order to adequately capture the given downstream task. For FNC-1 ARC this might not be observable since the heterogeneous data set requires a smaller learning rate in general. Both DistilBERT and ALBERT perform relatively bad for a learning rate of 1e-5 on the FNC-1 data set. Given the data set of the finetuning task is relatively homogeneous in its instances, lighter BERT-based frameworks require a larger learning rate than 1e-5 in order to achieve a better performance.

The general best performing learning rates are 1e-5 and 2e-5. It seems that a more cautious updating of the pretrained parameters is important which is a strong indication of how well the large pretrained models already perform. Li et al. (2020) note that "it is believed that adhering to the original hyperparameters for finetuning with small learning rate prevents destroying the originally learned knowledge or features." (Li et al., 2020, p.1). A small learning rate is seemingly the most important factor for correctly using large pretrained NLP models for downstream tasks. BERT and RoBERTa prefer a learning rate of 2e-5 and 1e-5 respectively and show the same tendencies in performance with respect to the learning rate for both data sets. Thus this recommendation is relatively stable. DistilBERT and ALBERT prefer a larger learning rate for the more homogeneous FNC-1 data set and a smaller learning rate for the more heterogeneous FNC-1 ARC data set. Later, it will become evident that ALBERT's better performance for the FNC-1 ARC data set can be explained largely by the improved prediction strength on the sparse category of *Disagree* instances. The more evenly distributed class labels of FNC-1 ARC have the biggest impact on BERT and ALBERT.

Looking at the overall performance, ALBERT performs the worst and even worse than BERT. This confirms the statements of Lan et al. (2019) who have reported that ALBERT can only outperform BERT for the xlarge and xxlarge variant. Surprisingly, DistilBERT can outperform BERT on the FNC-1 ARC data set and partly on the FNC-1 data set. When comparing the two different approaches of DAE versus AR, the best BERT-based model (RoBERTa) performs better than XLNet. XLNet still outperforms BERT, DistilBERT and ALBERT on both data sets and is thus the second-best overall model. However, XLNet bears the additional disadvantage of a much longer training time than any BERT-based model. While RoBERTa trains for around 59 minutes for one configuration, XLNet takes around 140 minutes which is more than twice the finetuning time of RoBERTa. All other BERT-based model train faster than RoBERTa. The reason why XLNet does not beat RoBERTa might originate from the specific advantage that XLNet introduces. Yang et al. (2019) criticize BERT for making the assumption that all masked tokens in a sequence are independent. By using the PLM objective, XLNet can avoid making such an assumption and is furthermore able to capture dependencies between tokens better than BERT-based architectures. The given task of stance detection is not a token-level but a segment-level task. The advantage of XLNet of better capturing the dependencies between tokens might thus be not of much use in this context. The overall winning model is RoBERTa with a batch size of 4, a sequence length of 512 and a learning rate of 1e-5 with a linear schedule for both data sets.

The evaluation done so far focused on the F_1-m metric. Since both data sets consist of unevenly distributed and heavily skewed class labels, the class-wise F_1 is now considered in order to gain further insights: In general, the performance improves for the extended data set FNC-1 ARC for all models. All models except for RoBERTa can drastically improve their prediction strength for the *Disagree* class which has the fewest training instances, precisely 1.7% (FNC-1) and 3.5% (FNC-1 ARC). RoBERTa can not push the performance on the extended data set as much on this category as other models. This could be interpreted as some sort of saturation effect, since using more evenly distributed data only helps to a certain degree. The FNC-1 ARC data set is still heavily skewed and in addition more heterogeneous than the FNC-1 data set. If the overall model architecture is already very powerful, as is the case for RoBERTa, the heterogeneity in the training instances might outweigh the positive factor of having more evenly distributed data.

XLNet can boost its performance for the extended data set more, in relative comparison to RoBERTa. This can be largely attributed to the boosted performance on the *Disagree* class. All models perform relatively well on the *Discuss* and extraordinarily well on the *Unrelated* class. The worst F_1-UNR values occur for ALBERT with 96.65 for the FNC-1 and 96.83 for the FNC-1 ARC data set.

RoBERTa is the strongest model since it also outperforms XLNet. This better performance is not attributable to one specific class, since RoBERTa performs stronger on all classes. When considering F_1-m, all models improve their performance when finetuned on more data, especially on the hardest to predict class *Disagree*.

In summary, the grid search showed a very strong performance of RoBERTa which also outperforms XLNet. This might be due to the specific FND task that is on a segment- rather than a token-level. In

	BERT		RoBERTa		DistilBERT		ALBERT		XLNet	
Metric	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC	FNC-1	+ ARC
F_1-m	**70.18**	**72.20**	**78.18**	**78.19**	**72.11**	**73.59**	**59.80**	**65.01**	**75.00**	**75.57**
F_1-AGR	60.31	63.48	70.69	70.57	61.95	65.29	53.19	53.97	68.00	68.57
F_1-DSG	41.76	48.28	56.15	58.92	45.09	50.46	13.21	34.07	49.47	53.69
F_1-DSC	80.36	78.82	86.78	84.16	82.83	80.22	76.16	75.18	83.73	81.43
F_1-UNR	98.28	98.22	99.10	99.09	98.58	98.38	96.65	96.83	98.80	98.60

Table 4: Model performances with respect to class-wise F_1 as well as F_1-m in comparison for FNC-1 and FNC-1 ARC. For better readability we indicate the columns for FNC-1 ARC just with "+ ARC".

addition to performing better, RoBERTa also trains much faster than XLNet. The learning rate is the most important hyperparameter which is in line with current research. For BERT and RoBERTa a clear recommendation for the learning rate can be drawn, while DistilBERT, ALBERT and XLNet exhibit their best performances for different values of the learning rate. The sequence length is not as important and dominated by the similarity of training instances within a data set. If the instances are more similar, the models manage finetuning well with shorter sequences. A more heterogeneous data set requires the model to consider more context and thus a longer sequences. As a learning rate schedule, a linear choice is sufficient and a cosine schedule is not necessary per se.

6 Conclusion

The overall conclusion that can be drawn from this work is how powerful and strong the evaluated architectures are for the examined task related to Fake News Detection. Even with minimal hyperparameter tuning and only finetuning for 3 epochs, the models already performed considerably well on both data sets. With respect to the considered hyperparameters the two most important conclusions are to not only finetune the classification and pooling layers that are stacked on top of the pretrained models[4]. Secondly, the most important hyperparameter is the learning rate. The recommendation of Goodfellow et al. (2016) to focus on the learning rate, when one has only time to address one single hyperparameter can be confirmed. At last, the models are relatively robust with respect to the learning rate schedule, the batch size, as long as it is adjusted to the learning rate and to a certain degree also the sequence length. Furthermore, the excessive pretraining approach of RoBERTa can outperform the permutation language model objective of XLNet.

Fake News detection itself is often time treated as a inherently binary task[5] (i.e. a document either contains Fake News or it does not). The underlying truth, however, might be more complicated, since most of the times a document can not unambiguously be assigned to either of these two classes. This is why our experiments on the FNC-1 data sets, which comprehend Fake News detection as a multi-class classification problem, might be more beneficial for this field of research compared to experiments on data sets with a binary target. We would not a go as far as concluding that these findings are without further ado transferable to other Fake News related data sets using different tasks or label sets, but nevertheless think that our findings can serve as starting points for further experiments in related fields.

Acknowledgements

We want to express our sincere gratitude to Christian Heumann (Ludwig-Maximilians-Universität) and Matthias Wissel (Capgemini Deutschland GmbH) for their help and constant support and advice during the process of conducting this research project. We would also like to thank the three anonymous reviewers for their insightful comments and their feedback on our work.

[4]For some models the pooling layer is already incorporated in the main architecture but still it does not suffice to only finetune the last pooling and classification layer.

[5]See for example the *Shared Task 2 (Fake news detection)* of the KDD 2020 TrueFact Workshop, where the texts are labeled either as "reliable" or "unreliable".

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Xishuang Dong, Uboho Victor, Shanta Chowdhury, and Lijun Qian. 2019. Deep two-path semi-supervised learning for fake news detection. *ArXiv*, 1906.05659.

William Ferreira and Andreas Vlachos. 2016. Emergent: a novel data set for stance detection. *Proceedings of NAACL-HLT 201*, pages 1163–1168.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Andreas Hanselowski, Avinesh PVS, Benjamin Schiller, Felix Caspelherr, Debanjan Chaudhuri, Christian M Meyer, and Iryna Gurevych. 2018. A retrospective analysis of the fake news challenge stance detection task. *arXiv preprint arXiv:1806.05180*.

Karen Hao. 2019. Training a single ai model can emit as much carbon as five cars in their lifetimes. https://www.technologyreview.com/s/613630/training-a-single-ai-model-can-emit-as-much-carbon-as-five-cars-in-their-lifetimes/. Accessed: 2020-02-18.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531.

Junaed Younus Khan, Md Khondaker, Tawkat Islam, Anindya Iqbal, and Sadia Afroz. 2019. A benchmark study on machine learning methods for fake news detection. *arXiv preprint arXiv:1905.04749*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *ArXiv*, 1412.6980.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotik, and Stefano Soatto. 2020. Rethinking the hyperparameters for fine-tuning. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

Tony Peng. 2019. The staggering cost of training sota ai models. https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/. Accessed: 2020-02-20.

Verónica Pérez-Rosas, Bennett Kleinberg, Alexandra Lefevre, and Rada Mihalcea. 2018. Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Dean Pomerleau and Delip Rao. 2017. Webpage of the fake news challenge 1 (fnc-1). http://www.fakenewschallenge.org/. Accessed: 2020-10-23.

Yuxiang Ren and Jiawei Zhang. 2020. Hgat: Hierarchical graph attention network for fake news detection. *ArXiv*, 2002.04397.

Natali Ruchansky, Sungyong Seo, and Yan Liu. 2017. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations Newsletter*, 19(1):22–36.

Craig Silverman. 2019. Emergent: A real-time tracker. http://www.emergent.info/. Accessed: 2020-10-23.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, 1910.03771.

Yang Yang, Lei Zheng, Jiawei Zhang, Qingcai Cui, Zhoujun Li, and Philip S. Yu. 2018. Ti-cnn: Convolutional neural networks for fake news detection. *ArXiv*, 1806.00749.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Seunghyun Yoon, Kunwoo Park, Joongbo Shin, Hongjun Lim, Seungpil Won, Meeyoung Cha, and Kyomin Jung. 2019. Detecting incongruity between news headline and body text via a deep hierarchical encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 791–800.

Appendix

A Finetuning time for different freezing techniques

	FNC-1			FNC-1 ARC		
	Freeze	No Freeze	Freeze Embed	Freeze	No Freeze	Freeze Embed
BERT	00:27:11 (sd = 4 sec)	01:01:45 (sd = 6 sec)	00:57:49 (sd = 6 sec)	00:32:40 (sd = 2 sec)	01:17:33 (sd = 6 sec)	01:12:37 (sd = 11 sec)
RoBERTa	00:28:24 (sd = 11 sec)	01:03:01 (sd = 7 sec)	00:58:59 (sd = 6 sec)	00:33:50 (sd = 5 sec)	01:18:49 (sd = 5 sec)	01:13:40 (sd = 5 sec)
DistilBERT	00:17:05 (sd = 1 sec)	00:35:19 (sd = 2 min)	00:34:08 (sd = 0 sec)	00:19:03 (sd = 2 sec)	00:43:30 (sd = 5 sec)	00:38:33 (sd = 25 sec)
ALBERT	00:20:01 (sd = 3 sec)	00:49:28 (sd = 3 sec)	00:47:36 (sd = 2 sec)	00:25:25 (sd = 3 sec)	01:03:43 (sd = 2 sec)	01:01:14 (sd = 3 sec)
XLNet	01:26:46 (sd = 50 sec)	02:26:01 (sd = 10 sec)	02:20:44 (sd = 18 sec)	01:49:07 (sd = 26 sec)	03:06:21 (sd = 53 sec)	02:58:49 (sd = 25 sec)

Table 5: Mean finetuning time in hh:mm:ss over three runs for the exploration step. For *Freeze* only the last projection and classification layers are updated. For *No Freeze* all layers are updated, while for *Freeze Embed* all embedding-specific layers are excluded from updating. The learning rate was kept fixed at 3e-5 with a linear schedule. The batch size was 8 with a sequence length of 512 tokens.