An Empirical study on Pre-trained Embeddings and Language Models for Bot Detection

Andres Garcia-Silva
Expert System
Calle Profesor Waksman 10
28036, Madrid, Spain
agarcia@expertsystem.com

Cristian Berrio
Expert System
Calle Profesor Waksman 10
28036, Madrid, Spain
cberrio@expertsystem.com

Jose Manuel Gomez-Perez
Expert System
Calle Profesor Waksman 10
28036, Madrid, Spain
jmgomez@expertsystem.com

Abstract

Fine-tuning pre-trained language models has significantly advanced the state of art in a wide range of downstream NLP tasks. Usually, such language models are learned from large and well-formed text corpora from e.g. encyclopedic resources, books or news. However, a significant amount of the text to be analyzed nowadays is Web data, often from social media. In this paper we consider the research question: How do standard pre-trained language models generalize and capture the peculiarities of rather short, informal and frequently automatically generated text found in social media? To answer this question, we focus on bot detection in Twitter as our evaluation task and test the performance of fine-tuning approaches based on language models against popular neural architectures such as LSTM and CNN combined with pre-trained and contextualized embeddings. Our results also show strong performance variations among the different language model approaches, which suggest further research.

1 Introduction

Recently, transfer learning techniques (Pan and Yang, 2010) based on language models have successfully delivered breaktrough accuracies in all kinds of downstream NLP tasks. Approaches like ULMFiT (Howard and Ruder, 2018), Open AI GPT (Radford et al., 2018) and BERT (Devlin et al., 2018) have in common the generation of pre-trained models learned from very large text corpora. The resulting language models are then fine-tuned for the specific domain and task, continuously advancing the state of the art across the different evaluation tasks and benchmarks commonly used by the NLP community.

Transfer learning approaches based on language models are therefore the NLP analogue to similar approaches in other fields of AI like Computer Vision, where the availability of large datasets like ImageNet (Deng et al., 2009) enabled the development of state of the art pre-trained models. Before language models, common practice for transfer learning in NLP was based on pre-trained context-independent embeddings. These are also learned from large corpora and encode different types of syntactic and semantic relations that can be observed when operating on the vector space. However, their use is limited to the input layer of neural architectures, and hence the amount of data and training effort necessary to learn a high performance task-related model is high since it is still necessary to train the whole network. Pre-trained language models, on the other hand, attempt to learn in the network structure the word inter-relations that can be leveraged during the fine-tuning step, usually by just learning a feed forward network for the specific task. The network architecture varies depending on the approach, including transformers (Vaswani et al., 2017) based on decoders, encoders and attention mechanisms, and bi-directional long-short term memory networks (Hochreiter and Schmidhuber, 1997).

Language models are usually learnt from high quality, grammatically correct and curated text corpora, such as Wikipedia (ULMFiT), BookCorpus (Open AI GPT), a combination of Wikipedia and BookCorpus (BERT) or News (ELMo). However, a very significant amount of the text to be analyzed nowadays is Web data, frequently from social media. The question that immediately arises is therefore whether such language models also capture the nuances of the short and informal language often found in social media channels.

In this paper we explore this question and empirically study how pre-trained embeddings and language models perform when used to analyze text from social media. To this purpose, we focus on bot detection in Twitter as evaluation task for two main reasons. First, the intrinsic relevance of the task for counteracting the automatic spreading of disinformation and bias on social media. Second, because in this context the gap, in terms of the quality and overall characteristics of the language used, between the corpora used to learn the language models and the task-specific text to be analyzed (automatically generated in a social media, micro-blogging context) can be particularly representative.

In our experiments, prior to evaluating the behavior of pre-trained language models, we test pre-trained embeddings as a baseline learned from general corpora, social media and informal vocabularies. We choose two popular NLP neural architectures for our binary classification task: Long Short Term memory networks (LSTM; Hochreiter and Schmidhuber, 1997) and convolutional networks (CNN; LeCun et al., 1998). We also pre-processed our Twitter dataset, observing a positive effect on our CNN and LSTM classifiers while on the other hand such effect was actually negative on some of the tested pre-trained language models.

In general, our results indicate that fine-tuned pre-trained language models outperform pre-trained and contextualized embeddings used in conjunction with CNN or LSTM for the task at hand. This shows evidence that language models actually capture much of the peculiarities of social media and bot language or at least are flexible enough to generalize during fine-tuning in such context. From the different language models we evaluated, Open AI GPT beats BERT (base) and ULMFit in the bot/no bot classification task, suggesting that a forward and unidirectional language model is more appropriated for social media messages than other language modeling architectures, which is relatively surprising. Nevertheless, the considerable experimentation we carried out has raised a number of additional questions that will need further research. During the workshop, we aim at sharing and discussing these questions with the participants.

The rest of the paper is structured as follows. Section 2 describes the state of the art about the different models and embeddings used in the experiments. Next, the experimental setup is presented in section 3, where the learning objective is defined as well as the dataset and the used pre-trained embeddings. Section 4 and 5 present the experiments using CNN and LSTM and different combinations of pre-trained, contextualized and dynamically generated embeddings learnt during training of the bot/no bot classification model. Then, section 6 describes the experiments with pre-trained language models. Finally, a discussion about the results is presented in section 7.

2 State of the Art

Mikolov's word2vec (Mikolov et al., 2013) approach that proposes an efficient way to learn embeddings by predicting words based on their context using negative sampling sparkled a new generation of embedding learning methods like GloVe (Pennington et al., 2014), Swivel (Shazeer et al., 2016) and FastText (Joulin et al., 2016). These embeddings capture semantic and syntactic relations between words that were mapped to vector operations in the multidimensional space. Nevertheless these approaches generate static, context-independent embeddings for words in the vocabulary. ELMo (Peters et al., 2018) overcome this limitation by generating representations for each word as a function of the input sentence. In addition, while pre-trained embeddings are used as input for neural networks, ELMo allows the end-task model to learn a contextualized linear combination of its internal representation.

Pre-trained embeddings are used as the first layer of models or as additional features to neural architectures. However as the models are initialized randomly a lot of training data was still required to get a high performance. To alleviate this problem ULMFiT (Howard and Ruder, 2018) proposes a transfer learning method that pre-trains a language model on a large corpus using 3-layer LSTM architecture that is then fine-tuned on the target task. In fact, the fine tuning is done at the language model level to reflect the target task distribution and at the task level.

In the same vein the Open AI Generative Pre-trained Transformer (GPT) (Radford et al., 2018) learns a language model on a large corpus using a multi-layer transformer decoder, and supervised fine-tuning to adapt the parameters to the target task. For tasks other than text classification the input is transformed into an ordered sequence that the pre-trained model can process. In contrast, BERT uses a bidirectional transformer (Devlin et al., 2018), also known as a transformer encoder, that learns representations jointly conditioned on left and right context in all layers. Similar to ELMo, ULMFiT and Open AI GPT which pre-train language models, BERT learning objective is a masked language model and a binarized next sentence prediction tasks. For a classification process all of the parameters of BERT and the classification layer are fine-tuned jointly to maximize the log-probability of the correct label.

Our contribution is an empirical study on the fitness of the fine-tuning of pre-trained language models when tested against text from social media and the target task is classification. We also show how pre-processing of the target task corpus can affect the performance of the pre-trained models, and compare them with the use of pre-trained and contextualized embeddings as inputs of CNN and BiLSTM for the classification task.

3 Experiments

To evaluate pre-trained language models with Twitter data we focus on the relevant problem of detecting bots in social media. Bots are automatic agents that publish information for a variety of purposes such as weather and natural hazards updates, and news, but also for spreading misinformation and fake news. In fact, as of 2017 it has been estimated that as 9% to 15% of twitter accounts are bots (Varol et al., 2017) which means that out of the 321 million active user accounts[1] the number of automatic agents range from 28 to 48 million.

Detecting bots can be addressed as a binary classification problem focusing only in the tweet textual content since our main target are language models, regardless of the other features that might be drawn from the social network, such user metadata, network features based on the follower and followee relations, and tweet and retweet activity.

3.1 Dataset

To generate a dataset of tweets generated by bots or humans we rely on an existing dataset of bot and human accounts published by Gilani et al. (2017). We create a balanced dataset containing tweets labelled as bot or human according to the account label. In total our dataset comprises 500,000 tweets where 279,495 tweets were created by 1,208 human accounts, and 220,505 tweets were tweeted from 722 bot accounts.

In this sample, bots tend to be more prolific than humans since they average 305 tweets per account which contrasts with the human average of 231. In addition, bots tend to use more URL (0.8313 URL per tweet) and hash tags (0.4745 hashtags per tweets) in their tweets than humans (0.5781 URL and 0.2887 hashtags per tweet). This shows that bots aim at maximizing visibility (hashtags) and to redirect traffic to other sources (URL). Finally, we found that bots display more egoistic behaviour than humans since they mention other users in their tweets (0.4371 user mentions per tweet) less frequently than humans (0.5781 user mentions per tweet).

3.2 Pre-trained embeddings

We use pre-trained embeddings to train the classifiers rather than doing it from scratch. We use pre-trained embeddings learned from Twitter itself, urban dictionary definitions to accommodate the informal vocabulary often used in the social network, and common crawl as a general source of information:

- glove.twitter[2]: 200 dimension embeddings generated from Twitter (27B tokens, 1.2M vocabulary) using GloVe (Pennington et al., 2014).
- word2vec.urban[3]: 100 dimension embeddings generated from Urban Dictionary definitions (568K vocabulary) using Word2Vec (Mikolov et al., 2013).
- fastText.crawl[4]: 300 dimension embeddings generated from Common Crawl (600B tokens, 1.9M vocabulary) using fastText (Mikolov et al., 2018)

4 CNN for text classification

We use convolutional neural networks (CNN; LeCun et al., 1998) for the bot detection task inspired by Kim's work (Kim, 2014) that showed how this architecture achieved good performance in several sentence classification tasks, and other reports like (Yin et al., 2017) that show good results in NLP tasks. The neural network architecture uses

[1]Twitter 4th quarter and fiscal year 2018 results: https://www.prnewswire.com/news-releases/twitter-announces-fourth-quarter-and-fiscal-year-2018-results-300791624.html

[2]https://nlp.stanford.edu/projects/glove/

[3]https://data.world/jaredfern/urban-dictionary-embedding

[4]https://fasttext.cc/docs/en/english-vectors.html

3 convolutional layers and a fully connected layer. Each convolutional layer has 128 filters of size 5, relu was used as activation function and max pooling was applied in each layer. The fully connected layer uses softmax as activation function to predict the probability of each message being written by a bot or a human. All the experiments reported hereinafter use a vocabulary size of 20k tokens, sequence size 200, learning rate 0.001, 5 epochs, 128 batch size, static embeddings unless otherwise stated, and 10-fold cross validation.

First we train the CNN classifier on our dataset using pre-trained embeddings and compare them with randomly generated embeddings. In addition, we pre-process our dataset using the same pre-processing script[5] that was applied when learning the GloVe Twitter embeddings. This pre-processing replaces, for example, URL, numbers, user mentions, hashtags and some ascii emoticons with the corresponding tags. Evaluation results are presented in table 1.

Embeddings	Dim.	Pre-proc.	Precision	Recall	F-Measure
random	300	No	0.7567	0.7551	0.7517
glove.twitter	200	No	0.7641	0.7618	0.7587
		Yes	0.7834	0.7790	0.7750
word2vec.urban	100	No	0.7122	0.7119	0.7075
		Yes	0.7601	0.7565	0.7522
fastText.crawl	300	No	*0.7679*	*0.7659*	*0.7627*
		Yes	**0.7858**	**0.7849**	**0.7829**

Table 1: Evaluation of CNN classifiers using random and pre-trained embeddings. Bold and italics are used for best classifiers using pre-processing or not pre-processing respectively.

In this setting, the best classifiers, according to the f-measure, is learned using fastText common crawl embeddings and the pre-processed dataset, followed by the classifier that uses GloVe Twitter embeddings also with pre-processing. In general pre-processing improves all the classifiers and evaluation metrics. Also notice that the CNN with word2vec urban dictionary embeddings without pre-processing underperformed the classifier that uses random embeddings, however when using pre-processing the metrics are better for the former.

4.1 Contextualized embeddings

In addition to static pre-trained embeddings we train CNN classifiers with dinamically-generated embeddings using ELMo. ELMo embeddings were generated from our dataset, however none of the trainable parameters (i.e., linear combination weights) were modified in the process. Due to the high dimension of these embeddings (dim=1024) we reduced the sequence size to 50 to avoid memory errors. Evaluation results, reported in table 2, shows that when the corpus was not pre-processed ELMo embeddings produced the best classifier, in terms of f-measure, when compared with classifiers learned from pre-trained embedddings and a dataset without pre-processing (see results in table 1 for comparison).

Embeddings	Dim	Preproc.	Precision	Recall	F-Measure
ELMo	1024	No	0.7766	0.7719	0.7675
		Yes	0.7859	0.7827	0.7798

Table 2: Evaluation of CNN classifiers using contextualized embeddings.

However, when the corpus was pre-processed the classifier learned from ELMo embeddings underperforms with respect to the best classifier learned from fastText common crawl embeddings, while outperforms the classifiers learned from GloVe and Urban dictionary. Nevertheless, in this setting ELMo embeddings produces the classifier with highest precision. Another important finding is that ELMo embeddings always generates the classifier with highest precision, regardless of data pre-processings.

4.2 Combining embeddings

We experiment by concatenating different pre-trained embeddings in the input layer of the CNN. Since fastText embeddings learned the best classifiers we pivot around them. Results in table 3 show that the best classifier is learned using fastText common crawl and GloVe Twitter embeddings with data pre-processing, and this classifier is better than any of the previous classifiers reported in tables 1 and 2.

Nevertheless, if we consider the results without pre-processing the combination of these embeddings with ELMo generates the best classifier, which is compatible with what we found above when ELMo embeddings help to learn the best classifier when the dataset was not pre-processed (see table 2). Similarly, this combination of embeddings helps to learn the classifier with highest precision regardless data pre-proccessing.

[5]https://nlp.stanford.edu/projects/glove/pre-process-twitter.rb

Embeddings	Pre-proc.	Precision	Recall	F-Measure
fastText.crawl+glove.twitter	No	0.7724	0.7704	0.7672
	Yes	0.7906	**0.7887**	**0.7862**
fastText.crawl+word2vec.urban	No	0.7598	0.7566	0.7526
	Yes	0.7826	0.7798	0.7767
fastText.crawl + glove.twitter	No	0.7675	0.7644	0.7606
+ word2vec_urban	Yes	0.7806	0.7782	0.775
fastText.crawl + glove.twitter	No	*0.7787*	*0.7771*	*0.7744*
+ ELMo	Yes	**0.7925**	0.7861	0.7816

Table 3: Evaluation of CNN classifiers using concatenations of pre-trained and contextualized embeddings. Bold and italics are used for best classifiers using pre-processing or not pre-processing respectively.

4.3 Dynamic and pre-trained embeddings

Another option to improve these classifiers is to allow the CNN to adjust dynamically the embeddings or part of them in the learning process. To do so, we generate 300 dimension embeddings initialized randomly and configure the CNN to make them trainable. In addition, we concatenate these random and trainable embeddings to the pre-trained and ELMo embeddings, which were not modified in the learning process. In this round of experiments we always use pre-processing since in the previous sections this option always improved the classifiers.

Table 4 shows that dynamic embeddings by themselves help to learn a classifier better than all the previous reported. Nevertheless, there exists the risk of over-fitting since the embeddings are tailored to the classification task, and that is why it makes sense to combine them with embeddings learned from other corpora. In this case, the combination of dynamic and ELMo embeddings generates the best classifier. Another interesting finding is that for the first time a classifier using word2vec urban dictionary is better than the others using GloVe twitter and fastText common crawl. We think that the reduced dimensionality of urban dictionary embeddings (100 dim) compared to Twitter and common crawl embeddings (200 dim and 300dim) allows the dynamic embeddings (300 dim) to influence more the learning process, and achieve better results.

Embeddings	Precision	Recall	F-Measure
dynamic	0.7956	0.7957	0.7950
dynamic + glove.twitter	0.8051	0.8042	0.8027
dynamic + fastText.crawl	0.8013	0.8016	0.8009
dynamic + word2vec.urban	0.8066	0.8053	0.8034
dynamic + ELMo	**0.8125**	**0.8097**	**0.8073**

Table 4: Evaluation of CNN classifiers using dynamic embeddings and pre-trained and contextualized embeddings using a pre-processed dataset

In addition, as shown in table 5 we evaluate different combination of the dynamic embeddings and concatenations of the pre-trained and contextualized embeddings. None of these attempts generate a better classifier than the one using the combination of dynamic embeddings and ELMo. Nevertheless, concatenating more embeddings never worsens the evaluation results, and most of the time improves them, with the exception of ELMo embeddings.

Embeddings	Precision	Recall	F-Measure
dynamic + glove.twitter + word2vec.urban	0.8067	0.8056	0.8041
dynamic + fastText.crawl +glove.twitter	0.8057	0.8045	0.8027
dynamic + fastText.crawl + glove .twitter + word2vec.urban	0.8092	0.8078	0.8060
dynamic + fastText.crawl + ELMo	0.8118	0.8093	0.8070
dynamic + fastText.crawl + glove .twitter+ELMo	0.8105	0.8088	0.8070
dynamic + fastText.crawl + glove .twitter+word2vec.urban+ELMo	0.8131	0.8096	0.8069

Table 5: Evaluation of CNN classifiers using dynamic embeddings and concatenations of to pre-trained and contextualized embeddings.

Figure 1 presents an overview of all the CNN classifiers evaluated so far using pre-processing sorted in descending order by f-measure. This figure shows how different classifiers were generated by using initially single pre-trained embeddings and combinations of them. The upper part of the figure is dominated by classifiers that use dynamic and pre-trained embeddings where ELMo embeddings are always involved.

5 Bidirectional long short term memory networks

In addition to CNN we test Long Short Term Memory networks LSTM (Hochreiter and Schmidhuber, 1997), a neural architecture that is also often used in NLP tasks (Yin et al., 2017). LSTM are sequential networks that are able to learn long-term dependencies. In our experiments we use a bidirectional LSTM that processes the sequence of text forward and backward to learn the model. The architecture of the BiLSTM comprises an embedding layer, the BiLSTM layer with 50 processing cells, and a fully connected layer that uses softmax as activation function to predict the probability of each message being written by a bot or a human. The rest of hyperparameters are set with the same values that we use for the CNN experiments.

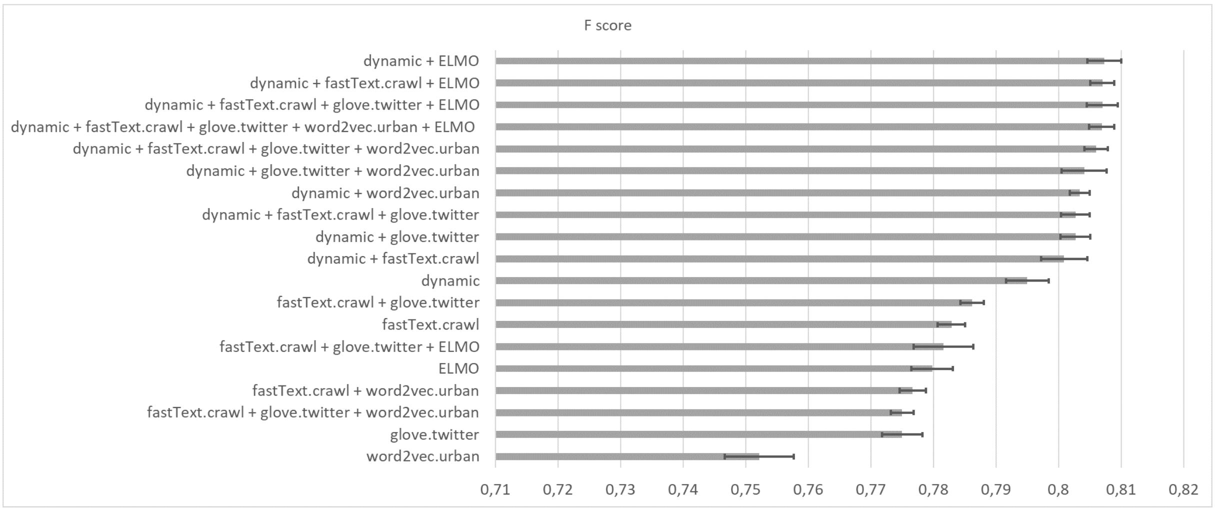

Figure 1: Evaluation of CNN classifiers learned from the single and concatenated pre-trained, contextualized and dynamic embeddings. The results are sorted in descending order by f-measure

In our experiments we test the embeddings combination that generates the best CNN classifiers: dynamic and ELMo embeddings, and also this combination enriched with fastText common crawl embeddings. Evaluation presented in table 6 shows that the best BiLSTM classifier learned from dynamic and ELMo emdeddings performs is very similar to the corresponding CNN. In fact, despite a slightly higher f-measure the individual values of precision and recall reported for the CNN are higher. In this experiment we do not find relevant differences between the CNN classifiers and their BiLSTM counterparts.

Embeddings	**Precision**	**Recall**	**F-Measure**
dynamic + ELMo	0.8095	0.8088	0.8074
dynamic+fastText.crawl + ELMo	0.8093	0.8087	0.8073

Table 6: Evaluation of BiLSTM classifiers using dynamic and pre-trained embeddings and a pre-processed corpus.

6 Pre-trained languages models and fine-tuning

In this section we present the evaluation results for the bot detection task using pre-trained language models and fine-tuning approaches. We follow the fine-tune procedures available for ULMFit[6], Open AI GPT[7], and BERT[8]. In all cases we use the default hyper-parameters:

- BERT base: 3 epochs, batch size of 32, and a learning rate of 2e-5
- Open AI GPT: 3 epochs, batch size of 8, and a learning rate of 6.25e-5
- ULMFiT: 2 epochs for the language model fine-tuning and 3 epochs for the classifier, batch size of 32, and a variable learning rate.

The classifiers evaluation results are presented in table 7. Considering f-measure the best classifier is learned by Open AI GPT, followed by BERT base model classifier. Transformer based approaches are more up to deal with social media messages. While Open AI GPT learns a classifier with highest recall, BERT base model does it with the highest precision. ULMFiT, on the other hand, achieves a high precision, although lower than the rest, and a low recall hence the low f-measure. Both, Open AI GPT and BERT base improve f-measure with respect to the best classifier learned previously by a BiLSTM using dynamic and ELMo embeddings (see table 6).

In addition, we evaluate how data pre-processing affects the pre-trained language models and fine-tuning approaches. Evaluation results presented in table 8 shows that while ULMFiT performance improves, Open AI GPT and BERT base worsen. Nevertheless, ULMFiT classifier is

[6]https://docs.fast.ai/text.html#Fine-tuning-a-language-model

[7]https://github.com/tingkai-zhang/pytorch-openai-transformer_clas

[8]https://github.com/google-research/bert#fine-tuning-with-bert

Pre-trained Language model	Precision	Recall	F-Measure
BERT base	**0.8572**	0.8213	0.8388
ULMFiT	0.8471	0.6902	0.7606
Open AI GPT	0.8567	**0.8546**	**0.8533**

Table 7: Pre-trained language models and fine-tuning without data pre-processing

still worse than Open AI and BERT base classifiers. Similarly to what we found above BERT base has the highest precisions while Open AI GPT the highest recall. Note that none of these classifiers beats the Open AI GPT learned from a non pre-processed dataset (see table 7).

Pre-trained Language model	Precision	Recall	F-Measure
BERT base	**0.8481**	0.7948	0.8206
ULMFiT	0.8096	0.7510	0.8123
Open AI GPT	0.8257	**0.8243**	**0.8229**

Table 8: Pre-trained Language models and fine tuning with data pre-processing

7 Discussion

In this paper we use a classification task to validate whether the improvement that transfer learning approaches based on fine-tuning pre-trained language models have brought to NLP tasks can be also achieved with social media text. The challenge for these models is that they have been learned from corpora like Wikipedia, News, or Books, where text is well written, grammatically correct and contextualized. On the other hand, social media messages are short and full of acronyms, hashtags, user mentions, urls, and mispellings. Our learning objective is detecting bots in Twitter messages since as automated agents the generated text is potentially different than the text sources used to pre-trained the language models.

We first present experimental results using classifiers trained with CNN and BiLSTM neural architectures along pre-trained, contextualized and dynamic embeddings. From the experiment results we conclude that using a concatenation of dynamically adjusted embeddings in the training process plus contextualized embeddings generated by ELMo helps to learn the best classifiers. Nevertheless, the models using ELMo embeddings exclusively were penalized when the training data was pre-processed. This was an unexpected result since ELMo works at the character level allowing it to work with unseen tokens like the tags that we use to replace the actual tokens in the messages.

Next, we fine-tune pre-trained language models generated with ULMFit, BERT base and Open AI GPT, showing that the last two approaches generate classifiers that outperform the best classifiers generated by the CNN and BiLSTM respectively, while ULMFit performance only improves over these classifiers when the data was pre-processed. In addition, BERT always learns the classifier with the highest precision while Open AI GPT learns the classifier with highest recall.

These results open many questions that need more research such as:

- Are unidirectional language models such as Open AI GPT more fitted for short and informal text?
- Is the bidirectional approach used in BERT contributing to the highest precision, while the masked tokens are decreasing its recall?
- Why does the BiLSTM approach used in UMLFiT perform better with pre-processed data in contrast to the other approaches or is this a result of the techniques used in the fine-tuning steps (gradual unfreezing, discriminative fine-tuning, and slanted triangular learning rates) ?

We expect to discuss these questions within the workshop to get more insights about the presented experimental work.

Acknowledgments

This work has been partially supported by LETSCROWD and The European Language Grid projects funded by the European Unions Horizon 2020 research and innovation programme under grant agreements No 740466 and No 825627, respectively. Special thanks to Ronald Denaux for discussion and technical support.

References

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

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

Zafar Gilani, Ekaterina Kochmar, and Jon Crowcroft. 2017. Classification of twitter accounts into automated agents and human users. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, pages 489–496, New York, NY, USA. ACM.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *IEEE Trans. on Knowl. and Data Eng.*, 22(10):1345–1359.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Noam Shazeer, Ryan Doherty, Colin Evans, and Chris Waterson. 2016. Swivel: Improving Embeddings by Noticing What's Missing. *arXiv preprint*.

Onur Varol, Emilio Ferrara, Clayton A. Davis, Filippo Menczer, and Alessandro Flammini. 2017. Online human-bot interactions: Detection, estimation, and characterization. In *ICWSM*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schtze. 2017. Comparative study of cnn and rnn for natural language processing.