GRUBERT: A GRU-Based Method to Fuse BERT Hidden Layers for Twitter Sentiment Analysis

Leo Horne* ETH Zurich hornel@ethz.ch

Matthias Matti* ETH Zurich mmatti@ethz.ch

Pouya Pourjafar* ETH Zurich ppouya@ethz.ch

Zuowen Wang* ETH Zurich wangzu@ethz.ch

Abstract

In this work, we introduce a GRU-based architecture called GRUBERT that learns to map the different BERT hidden layers to fused embeddings with the aim of achieving high accuracy on the Twitter sentiment analysis task. Tweets are known for their highly diverse language, and by exploiting different linguistic information present across BERT hidden layers, we can capture the full extent of this language at the embedding level. Our method can be easily adapted to other embeddings capturing different linguistic information. We show that our method outperforms well-known heuristics of using BERT (e.g. using only the last layer) and other embeddings such as ELMo. We observe potential label noise resulting from the data acquisition process and employ early stopping as well as a voting classifier to overcome it.

1 Introduction

With the rise of social media, Twitter has become an important and oft-used platform for sharing opinions and even doing politics. Furthermore, Twitter is a rich source of data collection since tweets are often closer to everyday spoken language than formally written texts. This has caused Twitter sentiment analysis (Kouloumpis et al., 2011) to become a well-established benchmark in the scientific community with practical applications such as predicting political preferences (Ansari et al., 2020). Moreover, it has piqued scientific interest in the field of natural language processing (NLP) as a source of learning informal languages. Colloquial language has many unique features as opposed to formal language, such as the use of slang words, misspellings, abbreviations, metaphors, sarcasm and context-dependent changes in meaning. Tweets also make extensive use of hashtags. This high volatility results in approaches well-suited to the analysis of formal texts giving sub-par performance on Twitter sentiment analysis.

The success of word embeddings such as Word2Vec and GloVe (Mikolov et al., 2013; Pennington et al., 2014) is based on their effectiveness in encoding semantic relationships between words. These were the first instances of word embeddings pre-trained on large corpora of text in an unsupervised fashion. However, one major drawback of these representations is that they do not account for the fact that a word can have different meanings based on its context, i.e. polysemy is not modeled. Additionally, words not in the dictionary cannot be easily taken into account in such models, which is a problem for tweets because of their frequent use of abbreviations, misspellings, and hashtags not present in the dictionary.

Following the idea of ELMo, more recent works expand unsupervised language models to a much larger scale by training them on large corpora of free text (Devlin et al., 2019; Radford, 2018; Radford et al., 2019; Brown et al., 2020). Unlike ELMo, which uses a multi-layer bi-LSTM (Hochreiter and Schmidhuber, 1997), these models are based on a multi-layer transformer architecture (Vaswani et al., 2017). Similarly to recurrent neural networks (RNNs) and LSTMs, transformers are designed to handle sequential data. However, they are entirely based on attention mechanisms (Bahdanau et al., 2014), which allow to train more powerful language models more efficiently. Transformer-based models have reached state-of-the-art performance in many NLP tasks (Liu et al., 2019b; Devlin et al., 2019; Lan et al., 2019; Radford, 2018; Radford et al., 2019; Brown et al., 2020). It has been shown that fine-tuning such pre-trained models is effective for many downstream tasks. (Howard and Ruder, 2018; Radford, 2018).

*All authors contributed equally to this work.

A recent major breakthrough in the realm of NLP was the advent of BERT (Devlin et al., 2019), which leveraged bi-directional transformers for language representations. One of the main advantages of using BERT for Twitter sentiment analysis is that it uses sub-tokens instead of a fixed per-word token. This makes it highly suitable for the Twitter dataset that often includes misspellings and slang words. Moreover, contextualized word representations can be extracted from hidden layers of the BERT model (Devlin et al., 2019). However, one of the main challenges is the question of how and which layers to use in order to fully optimize the performance for the downstream task (Kovaleva et al., 2019; Devlin et al., 2019). Different layers of the model have been shown to capture different linguistic information (Liu et al., 2019a). While earlier layers capture more low level information such as character-based features, the middle layers tend to capture syntactic information and later layers more semantic features (Jawahar et al., 2019). Hence, finding a good way to leverage the relevant information for a specific language task becomes an important problem.

Another research problem involving microblogging is the limitations rooted in automatic data collection. Various ways of automatically labelling twitter data have been developed, such as (Pak and Paroubek, 2010; Bifet and Frank, 2010) which use emoticons to label tweets. These processes inevitably introduce label noise in the training dataset, as is also stated in (Barbosa and Feng, 2010).

Problem setup Our training dataset[1] consists of 2.5 million labeled tweets, of which one half used to contain a positive smiley ":)" and the other half a negative smiley ":(". Those which previously contained a positive smiley are labeled as positive, and those which previously contained a negative smiley are considered to be negative. Since emoticons are not a perfect indicator for positivity or negativity of a tweet (especially due to phenomena like sarcasm and irony), we suspect that the dataset potentially contains label noise.

Given a tweet, our task is to predict its sentiment as either positive or negative. We split the dataset into a 70% training dataset and a 30% validation dataset. Furthermore, we use a separate test dataset consisting of 10 thousand unlabeled tweets (excluded from the aforementioned 2.5 million tweets) to report test accuracy.

Contributions We make the following contributions:

- We propose a novel architecture to create contextualized word representations from the hidden layers of the BERT model. Our architecture learns a combination of the hidden layers using gated recurrent units (GRUs) (Cho et al., 2014).
- We show that our method outperforms well-known heuristics for this task, e.g. concatenating the last four layers or using only the last hidden layer.
- We demonstrate that our proposed method is also applicable to other BERT-based models such as RoBERTa (Liu et al., 2019b).
- Using early stopping and a voting classifier, we prevent overfitting to possible label noise (which may result from many automatic data acquisition processes) and improve generalization of the model.

2 Models and Methods

In this section, we describe our pipeline for Twitter sentiment analysis.[2]

2.1 Pre-processing

In Section 1, we mention that tweets deviate from standard written texts in that they contain many abbreviations, slang, misspellings etc. not typically found in formal written text. Since most embeddings are trained on formal texts, we preprocess the datasets in an effort to make them conform more to the type of text the embeddings were trained on. The following data pre-processing steps are performed on the training set, validation set, and test set. We delete duplicate tweets to remove biases, remove excessive whitespaces from tweets and replace <user> (resulting from Twitter @mentions) and <url> (resulting from hyperlinks) by xxuser and xxurl respectively to avoid misinterpretations due to punctuation. Moreover, we use pyspellchecker (Barros, 2018) to correct misspelled words in each tweet.

2.2 Architectures

Our architecture aims at finding a way of combining different hidden layers of BERT such that the

[1]The dataset can be accessed at `https://www.kaggle.com/c/cil-text-classification-2020/data`

[2]The code can be found at `https://github.com/ZuowenWang0000/GRUBERT-A-GRU-Based-Method-to-Fuse-BERT-Hidden-Layers`

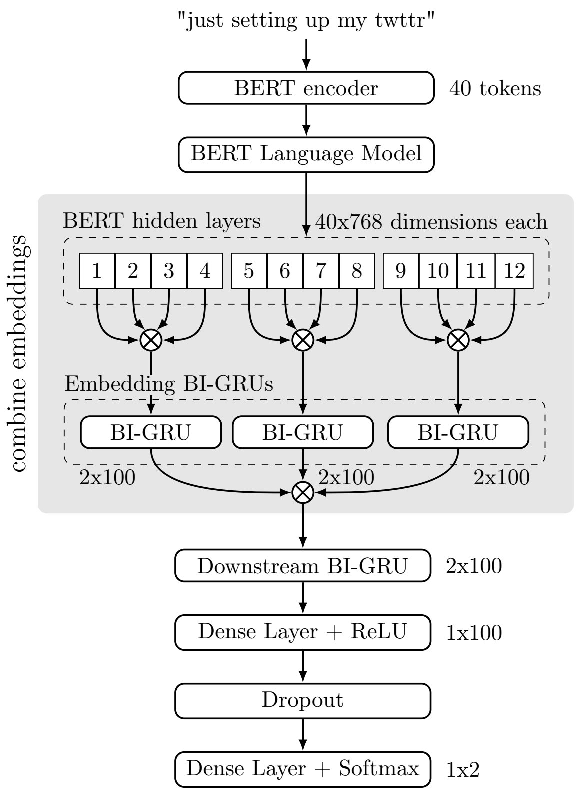

Figure 1: Illustration of the proposed architecture. The shaded part of the model indicates the combination of embeddings from hidden layers of the BERT model. The specific example shows the model *BERT-cat-3*, see Table 1. The symbol $\otimes$ represents the concatenation of tensors.

accuracy on the Twitter sentiment analysis task is increased compared to commonly used heuristics. Taking the average of the last four layers is a common heuristic which corresponds to a linear combination. Since the language used in tweets is very diverse, having a fixed way to combine the layers such as a linear combination may not leverage the full capabilities of BERT. For example, tweets including uncommon words might benefit more from information which is present in the earlier layers of BERT (character-based embeddings) whereas more formal tweets might benefit from the later layers. One possibility to overcome this challenge is to take the sequential information flow between subsequent layers into account via a recurrent unit. Hence, we opt for learning the combination of embeddings by utilizing gated recurrent units (GRUs) to capture the information flow from low level to high level features better. Moreover, we opt to first combine the BERT hidden layers in groups using a first layer of GRUs, then combine the output of the first layer of GRUs using another GRU. We hypothesize that grouping different layers together could be beneficial since the capacity of one bi-GRU could hinder its ability to capture the full information of the 12 layers. Thus, we divide the job by utilizing several bi-GRUs and assigning them grouped embeddings. We experiment with different number of bi-GRUs in order to find the best balance between incorporating information across layers and the capacity of one single bi-GRU.

An example of our model architecture is depicted in Figure 1. A tweet is first run through the BERT tokenizer, which prepares the inputs for the BERT model, i.e. tokenizes the input into sub-tokens, then embeds those sub-tokens. We heuristically clip tweets at 40 sub-tokens, since the 0.95-quantile of the number of words is 28. Shorter tweets are padded to the same length.

The tokenized tweet is then run through the BERT-base-uncased language model (Devlin et al., 2019; Wolf et al., 2019), which outputs 12 hidden layers of dimension 40×768. Each hidden layer can be interpreted as a sequence of 40 contextualized sub-token embeddings of dimension 1×768. Using a variable number of bi-GRUs, we combine multiple hidden layers into intermediate group embeddings. Each bi-GRU has a hidden state size of 100 for the forward and backward layer and creates a length 2×100-unit length embedding. We call these bi-GRUs *embedding bi-GRUs*. By concatenating the embeddings produced by embedding bi-GRUs, we obtain sub-token embeddings which contain information of all 12 layers, see the shaded area in Figure 1.

A further *downstream bi-GRU*—again with a hidden state size of 100 for both directions—is then run on the obtained embeddings. Its output is fed into a 200-unit fully-connected layer with rectified linear unit (ReLU) activation and dropout. A fully connected layer with 2 units followed by a softmax layer is added before the output is classified to either positive or negative. A cross-entropy loss is used for training the network.

Grouping hidden layers To determine the effect of group size on performance, we vary the combination of BERT hidden layers assigned to the embedding bi-GRUs. To keep the number of combinations of groups within reasonable limits, we assign the layers in uniformly sized groups to one embedding bi-GRU each, where the group size is a divisor of 12 (i.e. 1, 2, 3, 4 or 6). We further avoid shuffling the layers and only combine consecutive layers within groups.

Model	Hidden layer groups
BERT-cat-1	1-12
BERT-cat-2	1-6, 7-12
BERT-cat-3	1-4, 5-8, 9-12
BERT-cat-4	1-3, 4-6, 7-9, 10-12
BERT-cat-6	1-2, 3-4, 5-6, 7-8, 9-10, 11-12
BERT-share-c	see *BERT-cat-c*

Table 1: Listing of different models, which differ in the number of embedding bi-GRUs used to combine hidden layers. Each group of hidden layers is assigned to one GRU. BERT-cat-c has c embedding bi-GRUs ($c \in \{1, 2, 3, 4, 6\}$) while BERT-share-c has one embedding bi-GRU shared among different groups. For *BERT-share-c* we use the same grouping as in the *BERT-cat-c* models, with the difference that the lone GRU is shared among different embedding groups. Notice that BERT-cat-1 is equivalent to BERT-share-1.

Weight sharing We observe that some of our models benefit from sharing the weights of the embedding bi-GRUs. One possible reason for this could be that different groups of BERT hidden layers contain some of the same information. Indeed, the transformer architecture has skip connections, so some information from previous layers is passed onto the next layer. We further observe that weight sharing can prevent overfitting to some extent, as it implicitly induces regularization due to the fact that the degrees of freedom are more restricted, allowing the model to be trained for more iterations.

2.3 Training and implementation details

All models are implemented with PyTorch (Paszke et al., 2019). We use pre-trained BERT models and corresponding tokenizers from huggingface's transformers (Wolf et al., 2019) library.

Dense layers are initialized with Glorot initialization (Glorot and Bengio, 2010) and a dropout rate of 0.5 is used (Srivastava et al., 2014). We use the Adam optimizer (Kingma and Ba, 2015) with an initial learning rate of $1 \cdot 10^{-5}$, which is multiplied by 0.9 after each epoch. We further perform fine-tuning on the whole BERT model in every iteration in order to calibrate the embeddings with our dataset. For all experiments we use a batch size of 64 and train for 15 epochs in total. The hyperparameters are picked by a coarse grid search but due to computational resource constraint it is not exhaustive.

We train the models with one single GPU on a node equipped with an NVIDIA GeForce GTX 1080Ti and two 10-core Xeon E5-2630v4 processors. Each epoch takes approximately 1.5 hours for all models using BERT.

2.4 Preventing overfitting to label noise

Since the dataset is collected in an automated manner, there will inevitably be incorrectly labeled samples. Sarcasm and other rhetorics broadly exist in the tweets. Thus, :) and :(do not perfectly indicate the sentiment of the text. For example, *"grr .. ready for school .. i hate uniforms ! ! ugh we need our real clothes !"* is a picked sample from the training set where the label is positive but the ground truth is clearly negative.

Furthermore, the progression of the training also suggests the existence of noisy labels. In Figure 2 we show a typical training record when using a model trained exclusively on the last layer of BERT. The validation loss decreases when the validation accuracy increases at the beginning phase. However, after a turning point, the validation loss starts to rise while the validation accuracy keeps on going upwards, indicating that the classifier is less and less confident about its decisions as training progresses. This suggests that the classifier is overly considering data points with an incorrect label as opposed to correctly labeled data, making the model less confident on the latter. Notice that this phenomenon is most likely not due to a generalization problem, since the validation loss and accuracy are rising at the same time.

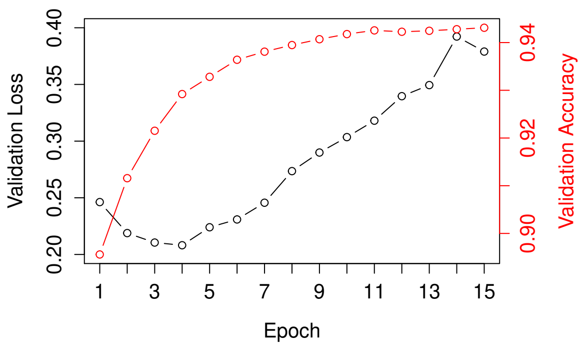

Figure 2: Validation loss and accuracy at different epochs for the model trained with the last layer of BERT with fine tuning. The validation loss reaches the lowest at the 4th epoch then starts increasing, while the validation accuracy is constantly rising until plateau. This indicates the existence of label noise in the training dataset.

In order to combat this problem, we apply early stopping and majority voting to ensure better robustness to label noise and generalization ability of the model. We assume that for the majority of the collected data samples, the sentiment corresponds to the label. Thus we aim to train the model to perform well on the data points in the test set whose labels match the actual sentiment.

Early stopping We split 30% from the preprocessed training set as a validation set, and we select the checkpoint with the lowest validation loss at the corresponding epoch, which is equivalent to early stopping on the criteria of the lowest validation loss. Li et al. (2019) suggest that overparameterized deep neural networks optimized by first-order gradient descent with early stopping are provably robust to label noise or corruption.

Majority voting Furthermore, a natural approach, which is also pointed out in (Frénay and Kabán, 2014), to alleviating this issue, and to make the model generalize better, is the use of voting classifiers. We ensemble several trained models and apply majority voting for the final prediction.

Our code framework can be easily extended to other multi-layer embeddings such as RoBERTa (Liu et al., 2019b). Therefore, we implement a voting classifier with BERT-base-uncased, RoBERTa and a multilingual BERT due to the presence of multilingual tweets in the dataset.

3 Results

In this section we first discuss the baselines we compare our models to and report in Tables 2 and 3 mean accuracies of three runs for each experiment, as well as the standard deviation. We report the accuracies of the method based on the test split of the dataset.

Major Baselines To assess the usefulness of the learned contextualized representations of our models, we implement a baseline architecture similar to the GRUBERT architecture as follows: after the embedding layer, there is a single bi-GRU (analogous to the embedding bi-GRUs from GRUBERT), followed by another bi-GRU (analogous to the downstream bi-GRU from GRUBERT), followed by a dense layer with ReLU activation, dropout, and dense layer with a softmax over the logits as in the GRUBERT architecture.

We also chose to compare the intermediate embeddings of our models with two BERT hidden layer combination heuristics used in (Devlin et al., 2019).

- *GloVe*: We use GloVe trained on Wikipedia 2014 and Gigaword 5 from Pennington et al. (2014).
- *ELMo*: We use ELMo embeddings (Peters et al., 2018) provided by the Flair NLP library (Akbik et al., 2019) (using AllenNLP (Gardner et al., 2017)).
- *BERT-last-layer*: This baseline uses the last hidden layer, i.e. layer 12.
- *BERT-last-four*: This baseline uses the concatenation of the last four hidden layers.

All baseline models are trained using the same hyperparameters as in Section 2.3, except for ELMo, which is trained with a learning rate of $1 \cdot 10^{-3}$. All baselines use one embedding bi-GRU followed by the rest of the downstream architecture, i.e. one embedding bi-GRU and a classifier.

BERT-cat and BERT-share Table 2 presents our results for the Twitter sentiment classification task using the models mentioned in Section 2. The *BERT-cat-2* and *BERT-cat-4* models outperform the equivalent parameter-sharing models, although *BERT-share-3* outperforms *BERT-cat-3*. It remains an open question why this is the case, although we suspect that it may be due to consecutive groups of four layers containing similar information to each other, while other consecutive groupings diverge more in the type of information contained in each grouping. Furthermore, we suspect that BERT-cat-2 outperforms the other BERT-cat models because BERT-cat models send different groups of layers through different bi-GRUs, thereby cutting the flow of information between layers. BERT-cat-2 only has two groups, so this cutting of information flow between layers is minimal. This hypothesis also explains why accuracy shows a downward trend as the number of groups in BERT-cat models is increased.

We also observe that certain configurations of GRUBERT outperform other commonly used embeddings such as GloVe, ELMo, as well as other common ways of using BERT embeddings, such as using only the last layer or concatenating the last four layers.

Table 3 validates our idea of using a GRU to capture the fact that different tweets may each benefit from different BERT layers by replacing the first layer of GRUs with fully connected linear layers. A linear layer always combines layers in the same way, so different tweets are always associated with the same combination of the embedding

Model	Mean Accuracy (Std. Dev.)
GloVe	83.52
ELMo	86.44
BERT-last-layer	89.06 (0.05)
BERT-last-four	89.27 (0.15)
BERT-cat-2	**89.43 (0.02)**
BERT-cat-3	89.21 (0.13)
BERT-cat-4	89.22 (0.26)
BERT-cat-6	89.05 (0.21)
BERT-share-1	89.37 (0.19)
BERT-share-2	89.04 (0.20)
BERT-share-3	**89.66 (0.18)**
BERT-share-4	89.14 (0.35)
BERT-share-6	89.02 (0.24)

Table 2: Experiment results for baselines, BERT-cat models and BERT-share models. The mean accuracy is computed over several runs of the model and evaluated on the test set. Note that *BERT-cat-1* is equivalent to *BERT-share-1*, hence it is omitted from the table. The GloVe and ELMo baselines are presented without standard deviation due to computational resource constraints and since they are significantly worse than BERT-based approaches.

layers, as opposed to a GRU, which can generate different combinations of layers for different tweets due to the recurrent information flow, and is therefore more context-sensitive. Table 3 shows that we obtain a higher accuracy using GRUs.

Model	Mean Acc. (Std. Dev.)
BERT-share-3-linear	89.43 (0.17)
BERT-share-3	**89.66 (0.18)**

Table 3: Comparison between linear layers for layer group combining vs GRUs. *BERT-share-3-linear* is equivalent to *BERT-share-3*, but with the first layer of GRUs replaced by fully connected linear layers.

Our method can be easily extended to other multi-layer embeddings such as RoBERTa (Liu et al., 2019b), leaving possibilities of adapting ensemble methods. We evaluate the final model by implementing our technique on top of a RoBERTa (Liu et al., 2019b) model and doing an ensemble with various BERT-share-3 models trained: (1) as described in Section 2, (2) on the full dataset, (3) using multilingual BERT embeddings, (4) with a weight decay of $1 \cdot 10^{-5}$, (5) using RoBERTa embeddings, as well as (6) a BERT-share-4 and (7) a BERT-share-6 model, both trained with RoBERTa embeddings. Using this technique we reach a final test score of 90.94%.

4 Discussion

We show empirically that GRUBERT is superior to standard embeddings for the task of Twitter sentiment analysis. However, our model is not easily interpretable and does not allow deeper insights into the BERT hidden layers, as is made apparent by the fact that we cannot draw concrete conclusion about why weight sharing gives a boost to certain groupings but not to others. In future work we would like to find interpretable combinations of the BERT layers for different task, to better understand the linguistic features present in each hidden layer.

As another item of future work, the effect of each component on the overall architecture should be examined more closely.

Furthermore, the effectiveness of our approach for other NLP tasks remains to be tested. However, our architecture can easily be used as a plug-in module for other multi-layer embeddings and downstream models, allowing the effectiveness to be easily examined by future work.

5 Conclusion

We have shown that a dynamic way of combining the BERT hidden layers using GRUs can lead to performance benefits in the case of irregular and plastic language found in tweets. We further experimented with different ways of combining the embeddings and observed that weight sharing can benefit the training process by implicitly inducing regularization and restricting the model complexity. We used early stopping as well as voting classifiers to address the label noise problem inherent in the automatic data collection process Using these findings, we develop a framework for the problem of machine-labeled Twitter sentiment analysis which makes use of an ensemble of different GRUBERT models to combat label noise.

Acknowledgements

We thank the Data Analytics Lab at ETH Zurich for providing computing infrastructure. We also thank them, in addition to our mentor Shuhei Kurita and the anonymous reviewers, for valuable feedback.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Mohd Zeeshan Ansari, Areesha Fatima Siddiqui, and Mohammad Anas. 2020. Inferring political preferences from twitter.

Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Coling 2010 - 23rd International Conference on Computational Linguistics, Proceedings of the Conference*, volume 2, pages 36–44.

Tyler Barros. 2018. pyspellchecker. Available online: https://github.com/barrust/pyspellchecker. Consulted 2020-07-27.

Albert Bifet and Eibe Frank. 2010. Sentiment knowledge discovery in twitter streaming data. pages 1–15.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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. Association for Computational Linguistics.

Benoît Frénay and Ata Kabán. 2014. A comprehensive introduction to label noise. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the 2011 International Conference on Web and Social Media (ICWSM)*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the Dark Secrets of BERT. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

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.

Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. 2019. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, Workshop Track Proceedings*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of LREC*, volume 10.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'é Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

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)*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Shikhar Vashishth, Prateek Yadav, Manik Bhandari, Piyush Rai, Chiranjib Bhattacharyya, and Partha P. Talukdar. 2018. Graph convolutional networks based word embeddings. *CoRR*, abs/1809.04283.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

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*, abs/1910.03771.

A Appendices

A.1 Number of Words per Tweet

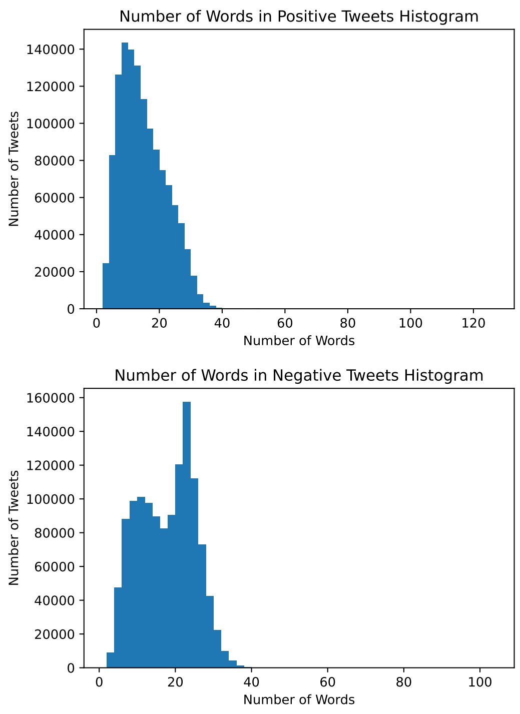

Figure 3

Figure 3 illustrates histograms of the number of words in the original data sets with positive and negative tweets.

Dataset	Positive Tweets	Negative Tweets
#Tweets	1.25mio	1.25mio
Mean	14.34	17.14
Std. Dev.	7.18	7.31
Max	128	104
Min	1	1
Quantiles		
0.05	5	6
0.25	5	6
0.5	13	18
0.75	19	23
0.95	28	28

Table 4: Statistics about number of words in tweets.

A.2 Additional Baselines

In this section, we present additional baselines using context-sensitive word embeddings such as ELMo (Peters et al., 2018), Flair (Akbik et al., 2018) and a graph convolution network based embedding SynGCN (Vashishth et al., 2018). We also try different stackings of these embeddings then feed them into the embedding bi-GRU followed by the downstream architecture. The training schedule is the same as mentioned in section 2.3. The following embeddings are used (Suffix -ft indicates with fine-tuning):

- *SynGCN*: We the pretrained SynGCN embedding from (Vashishth et al., 2018).
- *GloVe-SynGCN*: We stack the GloVe embedding used in 2 with the SynGCN embedding.
- *ELMo-mix*: On top of GloVe-SynGCN, we stack the ELMo embedding same as used in table 2. We use starting learning rate $1 \cdot 10^{-4}$ (choosed by grid search) with decay by multiplying 0.9 after every epoch.
- *Flair-mix*: On top of GloVe-SynGCN, we stack the Flair embedding same as used in table 2. For *Flair-mix-ft* fine-tuning was used.

Model	Accuracy
SynGCN	83.48
GloVe-SynGCN	85.44
ELMo-mix	86.30
Flair-mix	86.44
Flair-mix-ft	87.16

Table 5: Results from additional baselines.