GRUBERT: A GRU-Based Method to Fuse BERT Hidden Layers

Matthias Matti* mmatti@ethz.ch
Pouya Pourjafar* ppouya@ethz.ch
Zuowen Wang* wangzu@ethz.ch
Leo Horne* hornel@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. Additionally, we discover noisy labels inherent in the Twitter dataset acquisition process and employ early stopping and a voting classifier to overcome this.

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 [18] to become a well-established benchmark in the scientific community with practical applications such as predicting political preferences [3]. 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, typically by concatenating several words or abbreviations into one string preceded by the character “#”. This high volatility results in approaches well-suited to the analysis of formal texts giving sub-par performance on Twitter sentiment analysis.

Previous advances in natural language processing are based on neural networks that use dense vector representations of words, so-called word embeddings. In such models, the embedding layer contains a mapping from a dictionary of words to vectors. The success of word embeddings such as Word2Vec and GloVe [24, 26] 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.

ELMo [27] introduced contextual word embeddings using a language modeling approach to overcome these shortcomings. Its contextualized word representations use vectors that are a combination of the internal layers of a bidirectional long short-term memory (bi-LSTM) network that is trained to predict the next word in a sequence of words on a large text corpus. These word embeddings are included as additional features in task specific architectures to solve downstream tasks. Additionally, ELMo forms word embeddings based on characters, which means that it can generate meaningful embeddings for out-of-vocabulary words.

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 [6, 8, 28, 29]. Unlike ELMo, which uses a multi-layer bi-LSTM [13], these models are based on a multi-layer transformer architecture [32]. Similarly to recurrent neural networks (RNNs) and LSTMs, transformers are designed to handle sequential data. However, they are entirely based on attention mechanisms [4], which allow to train more powerful language models more efficiently. Transformer-based models have reached state-of-the-art performance in many NLP tasks [6, 8, 20, 23, 28, 29]. It has been shown that fine-tuning such pre-trained models is effective for many downstream tasks. [14, 28].

A major breakthrough was the advent of BERT [8], which leveraged bi-directional transformers for language representations. BERT has achieved great performance on a variety of language tasks [9, 15]. One of the main advantages of using BERT 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, ELMo-like contextualized word representations can be extracted from hidden layers of the BERT model [8]. 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 [8, 19]. Different layers of the model have been shown to capture different linguistic information [22]. 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 [16]. Hence, finding a good way to leverage the relevant information for a specific language task becomes an important problem.

Problem setting The provided training dataset 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), the dataset contains a significant amount of label noise. Given a tweet, our task is to predict its sentiment.

*All authors contributed equally to this work.

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) [7].
- We discover that fine-tuning the BERT model on the Twitter sentiment classification task provides an additional boost to our accuracy.
- 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 [23].
- Using early stopping and a voting classifier, we gain robustness against label noise and improve generalization of the model.

2 Models and Methods

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

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 written 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 [5] to correct misspelled words in each tweet. Manual inspection of the resulting output shows that although the spell-checker is not always 100% accurate, the amount of nonsensical spelling corrections is reasonably low.

2.2 Architectures

Our architecture aims at finding a good way of combining different hidden layers of BERT. Taking the average of the last four layers is a common heuristic which corresponds to a linear combination of the hidden layers. 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 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.

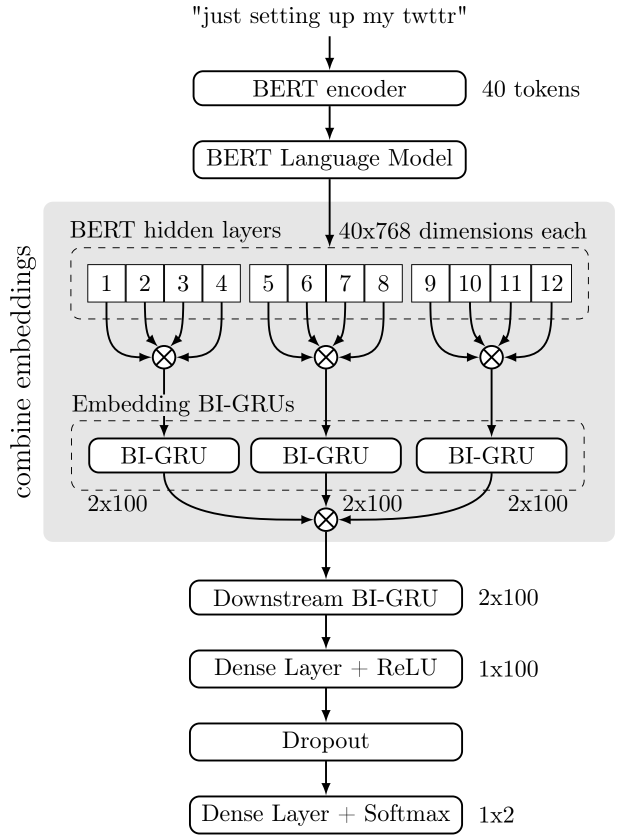

Figure 1: Illustration of the proposed architecture. The shaded part of the model indicates the combination of embeddings from hidden layers of the BERT model. The specific example shows the model *BERT-cat-3*, see table 1. The symbol ⊗ represents the concatenation of tensors.

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 (see Appendix A for further analysis). Shorter tweets are padded to the same length.

The tokenized tweet is then run through the BERT-base-uncased language model [8, 33], 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*x*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 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 a groups.

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

Table 1: Listing of the 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. For *BERT-share-c* we use the same grouping as in the *BERT-cat-c* models where $c \in \{1, \ldots, 6\}$, with the difference that the lone GRU is shared among different embedding groups. Notice that BERT-cat-1 is equivalent to BERT-share-1. Both only have one embedding bi-GRU.

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. We further observe that weight sharing can prevent overfitting to some extend, 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 [25]. We use pretrained BERT models and corresponding tokenizers from huggingface's transformers [33] library.

Dense layers are initialized with Glorot initialization [12] and a dropout rate of 0.5 is used [30]. We use the Adam optimizer [17] with an initial learning rate of $1 \cdot 10^{-5}$, which is multiplied by 0.9 after each epoch. We further perform finetuning on the whole BERT model in every iteration in order to calibrate the embeddings with our dataset. This proves to be a crucial step with significant improvement in the performance. 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, i.e. when :) exists in the text it is labeled as positive and :(stands for negative, there will inevitably be incorrectly labeled samples. Sarcasm and other rhetorics broadly exist in the twitter posts. 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 (over)fitting to noisy labels.

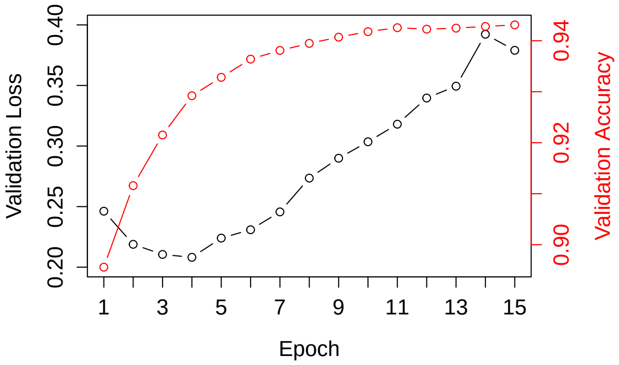

Figure 2: Validation loss and accuracy at different epochs for the model trained with the last layer of BERT with fine tuning. The validation loss reaches the lowest at the 4-th epoch then starts increasing, while the validation accuracy is constantly rising until plateau. This indicates the existence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

Early stopping We split 20% 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 [21] suggest that overparameterized deep neural networks optimized by first-order gradient descent with early stopping is provably robust to label noise or corruption.

Majority voting Furthermore, a natural approach, which is also pointed out in [10], 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 [23]. 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 table 2 and table 3 mean accuracies of three runs for each experiment, as well as the standard deviation. The accuracies (in percentage) correspond to the public score from the Kaggle system. The Kaggle private score was not available at the time of writing.

Major Baselines To assess the usefulness of the learned contextualized representations of our models, the architectures for the baselines differ from those described in section 2 only in the part where hidden layers are combined, i.e. in the shaded area of figure 1. We chose to compare the intermediate embeddings of our models with two hidden layer combination heuristics used in [8].

- *GloVe*: We use GloVe trained on Wikipedia 2014 and Gigaword 5 from [26].
- *ELMo*: We use ELMo embeddings [27] provided by the Flair NLP library [1] (using AllenNLP [11]).
- *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. We also test other ways of combining various embeddings and leave the results in Appendix B.

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.

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

Model	Mean Accuracy (Standard Deviation)
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. Note that *BERT-cat-1* is equivalent to *BERT-share-1*, hence it is omitted from the table. The GloVe, Flair, and ELMo baselines are presented without standard deviation due to computational resource constraints and since they are significantly worse than BERT-based approaches.

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 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 Accuracy (Std. Deviation)
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 directly applied to other BERT based architectures. We evaluate the final model by implementing our technique on top of a RoBERTa [23] 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 Kaggle public 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.

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. Through some data analysis, we observed that the labeling of our dataset is very noisy and used early stopping as well as voting classifiers to combat the noisy label problem. Using these findings, we develop a final solution for the task of Twitter sentiment analysis which makes use of an ensemble of different GRUBERT models to combat label noise and reach a test accuracy of 90.94%.

References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 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, June 2019. Association for Computational Linguistics.

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

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

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 09 2014.

[5] Tyler Barros. pyspellchecker, 02 2018. Available online: https://github.com/barrust/pyspellchecker. Consulted 2020-07-27.

[6] 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. Language models are few-shot learners. 2020.

[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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 2019. Association for Computational Linguistics.

[9] Junchao Dong, Feijuan He, Yunchuan Guo, and Huibing Zhang. A commodity review sentiment analysis based on bert-cnn model. *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, pages 143–147, 2020.

[10] Benoît Frénay and Ata Kabán. A comprehensive introduction to label noise. In *ESANN*, 2014.

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

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *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, 13–15 May 2010. PMLR.

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

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

[15] Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on wikisql with table-aware word contextualization. *ArXiv*, abs/1902.01069, 2019.

[16] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 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, July 2019. Association for Computational Linguistics.

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

[18] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! 01 2011.

[19] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 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)*, 2019.

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

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

[22] Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 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, June 2019. Association for Computational Linguistics.

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

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[25] 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. 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., 2019.

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

[27] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, 2018.

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

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

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

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

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

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

Appendices

A Number of Words per Tweet

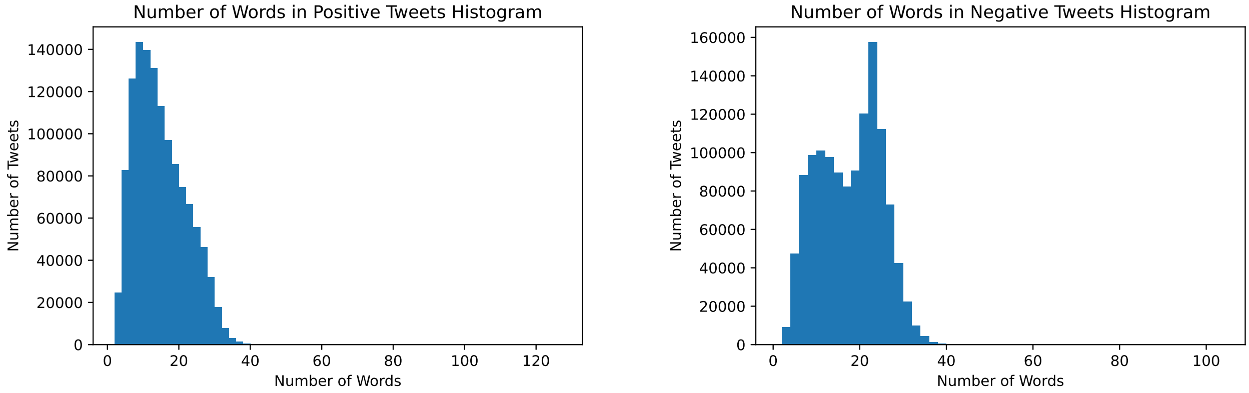

Figure 3: Histograms of the number of words in the original data sets with positive and negative tweets.

Dataset	#Tweets	Mean	Std. Dev.	Max	Min	Quantiles				
						0.05	0.25	0.5	0.75	0.95
Positive Tweets	1.25 mio	14.34	7.18	128	1	5	9	13	19	28
Negative Tweets	1.25 mio	17.14	7.31	104	1	6	11	18	23	28

Table 4: Statistics about number of words in tweets.

B Additional Baselines

In this section, we present additional baselines using context-sensitive word embeddings such as ELMo [27], Flair [2] and a graph convolution network based embedding SynGCN [31]. 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 [31].
- *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	*SynGCN*	*GloVe-SynGCN*	*ELMo-mix*	*Flair-mix*	*Flair-mix-ft*
Accuracy	83.48	85.44	86.30	86.44	87.16

Table 5: Results from additional baselines.

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

GRUBERT: A GRU-Based Method to Fuse BERT Hidden Layers

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Matti	Matthias
Pourjafar	Pouya
Wang	Zuowen
Horne	Leo

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the '[Citation etiquette]' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date

Zürich, 31.07.2020

Signature(s)

Zuowen Wang

L. Ho

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.