Reed at SemEval-2020 Task 9: Fine-Tuning and Bag-of-Words Approaches to Code-Mixed Sentiment Analysis

Vinay Gopalan
Reed College
gopalanvinay@gmail.com

Mark Hopkins
Reed College
hopkinsm@reed.edu

Abstract

We explore the task of sentiment analysis on Hinglish (code-mixed Hindi-English) tweets as participants of Task 9 of the SemEval-2020 competition, known as the SentiMix task. We had two main approaches: 1) applying transfer learning by fine-tuning pre-trained BERT models and 2) training feedforward neural networks on bag-of-words representations. During the evaluation phase of the competition, we obtained an F-score of 71.3% with our best model, which placed 4^{th} out of 62 entries in the official system rankings.

1 Introduction

The Internet today has a vast collection of data in various forms, including text and images. A big part of this data comes from various social media platforms, which enable users to share thoughts about their daily experiences. The millions of tweets, updates, location check-ins, likes, and dislikes that users share every day on different platforms form a large bank of opinionated data. Extracting sentiment from this data, though immensely useful, is also challenging, giving rise to the NLP task known as *sentiment analysis*. Several models have been proposed to perform this task over the years, such as those built on top of the Recursive Neural Tensor Network (Socher et al., 2013) or the more recent BERT model (Devlin et al., 2018). Although most of these language technologies are built for the English language, a lot of Internet data comes from multilingual speakers who combine English with other languages when they use social media. Thus, it is also important to study sentiment analysis for this so-called 'code-mixed' social media text. In this paper, we explore sentiment analysis on code-mixed Hinglish (Hindi-English) tweets, specifically as a participant in Task 9 of SemEval-2020 (Patwa et al., 2020).

We had two main strategies. We began by fine-tuning pre-trained BERT models to our target task. During this initial phase, we observed that the accuracies yielded by `bert-base`, `bert-multilingual` and `bert-chinese` on the validation data were approximately the same. This made us hypothesize that the pre-trained weights were mostly unhelpful for the task. To test this, we attempted to recreate the BERT fine-tuning results only using feedforward neural networks trained on a bag-of-words (BoW) representation.

We found that the the results of fine-tuning `bert-large` (24 layers) could be approximated by a 2-layer BoW feedforward neural network, as our best-performing fine-tuned model had an accuracy of 63.9% and our best-performing bag-of-words model had an accuracy of 60.0% on the validation corpus. In the evaluation phase of the competition, our fine-tuned model had an F-score of 69.9% without bagging (Breiman, 1996) and 71.3% with bagging. Once the official system rankings were published on April 6, 2020[1], our best submission (CodaLab username: `gopalanvinay`) placed 4th out of 62 entries. All code needed to recreate our results can be found here[2].

[1] see https://competitions.codalab.org/competitions/20654#learn_the_details-results

[2] see https://github.com/gopalanvinay/thesis-vinay-gopalan

2 Background

The organizers of SentiMix simultaneously organized a Hinglish and an analogous Spanglish (code-mixed Spanish-English) task (Patwa et al., 2020). We participated in the Hinglish track.

The task was to predict the sentiment of a given code-mixed tweet. Entrants were provided with training and validation data. These datasets were comprised of Hinglish tweets and their corresponding sentiment labels: positive, negative, or neutral. Besides the sentiment labels, the organizers also provided language labels at the word level. Each word is tagged as English, Hindi, or universal (e.g. symbols, mentions, hashtags). Systems were evaluated in terms of *precision*, *recall* and F_1-*measure*.

All data was provided in tokenized CoNLL format. In this format, the first line (prefaced with the term "meta") provides a unique ID and the sentiment (positive, negative or neutral) of the tweet. Each tweet is then segmented into word/character tokens and each token is given a language ID, which is either *HIN* for Hindi, *ENG* for English and *O* if the token is in neither language. Below is an example for a positive-sentiment tweet with id 173:

meta	173	positive
@	O	
BeingSalmanKhan	Eng	
It	Eng	
means	Eng	
sidhi	Hin	
sadhi	Hin	
ladki	Hin	
best	Eng	
couple	Eng	

The data contains tokens like usernames (e.g. @BeingSalmanKhan) and URLs, which we deemed unhelpful for sentiment analysis. Prior to any training, we converted the CoNLL format into a simpler format, consisting of the reconstituted tweet (omitting usernames and URLs) and the sentiment. For the above example, the converted tweet would be:

It means sidhi sadhi ladki best couple (positive)

Note that we ignore the token-level language ids.

3 Experimental Setup

For the experimental results reported in this paper (except for the final system results, which use the official competition test set), we trained our systems using the provided training set of 14K tweets and report results on the provided validation set of 3K tweets. Since the task is a relatively balanced three-way classification task, we used simple accuracy as our hillclimbing metric.

We used PyTorch v1.2.0 [3] and Python 3.7.1[4]. All additional details needed to replicate our results are provided in the README of our project's code repository [5].

4 Fine-tuned BERT

For our baseline experiments, we used the PyTorch-based implementation of BERT (Devlin et al., 2018) provided by Huggingface's `transformers` package (Wolf et al., 2019). We adapted the `SST-2` task of the `run_glue.py` script. The original task was configured for the binary classification of sentences from the Stanford Sentiment Treebank (Socher et al., 2013), as administered by the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). We adapted the task for three-way (positive, negative, neutral) classification.

[3] see https://pytorch.org/docs/1.2.0/

[4] see https://docs.python.org/3/

[5] see https://github.com/gopalanvinay/thesis-vinay-gopalan

pretrained model	accuracy (%)
bert-base-cased	62.2
bert-large-cased	63.3
bert-base-multilingual-cased	62.3
bert-base-chinese	61.0

Table 1: Results from fine-tuning BERT using various pre-trained models.

4.1 Experiment: Varying the Base Model

In our first experiment, we compared several pre-trained models:

- `bert-base-cased`: A 12-layer transformer with token embeddings of size 768, trained by Google on English data.
- `bert-large-cased`: A 24-layer transformer with token embeddings of size 1024, trained by Google on English data.
- `bert-base-multilingual-cased`: A 12-layer transformer with token embeddings of size 768, trained by Google on the Wikipedia dumps from 104 languages, including Hindi and English.

For these experiments, we used the default parameters provided by the Huggingface training script. Our hypothesis was that the `bert-base-multilingual-cased` model, which is simultaneously pre-trained on both Hindi and English, would be more effective[6] for Hindi-English tweet classification, but this did not turn out to be the case. In fact, there was little difference between the English-only `bert-base-cased` model and the `bert-base-multilingual-cased` model.

This observation suggested a new hypothesis, which was that pretraining was possibly not at all helpful for the code-mixed domain, and that the Transformer was simply learning to classify "from scratch," starting from the unhelpful weight initializations provided by the pretrained models. To test this hypothesis, we performed fine-tuning with the `bert-base-chinese` model, a 12-layer transformer with token embeddings of size 768, trained by Google on traditional and simplified Chinese text. The final line of Table 1 shows this result, which is surprisingly close to the models trained using English data, suggesting that the pre-training has relatively little impact on the task performance.

5 Bag-of-Words Models

The fine-tuning results suggested two possibilities:

1. Even though the pre-training makes little difference, the complex Transformer model still learns deep and interesting patterns to achieve an accuracy in the low sixties.
2. The Transformer model is only learning simple heuristics that could be just as easily learned by simpler models.

To distinguish between these possibilities, we attempted to replicate the performance of the fine-tuned systems using classical bag-of-words (BoW) models (McTear et al., 2016).

To create a BoW system, we iterated through the training data and stored the frequency of each word in the corpus. We then created a *vocabulary* $\mathbf{V} = \{v_1, v_2, ..., v_n\}$, which was the set of words that appeared with frequency at least K in the training data, for some positive *frequency threshold* $K \in \mathbb{N}$. After removing stop words from the vocabulary, we used this vocabulary to transform each tweet into an n-length vector whose j^{th} element equaled 1 if word $v_j \in \mathbf{V}$ appeared in the tweet (and 0 otherwise).

We then trained a simple classifier using these BoW vector representations of the tweets. For our classifier, we used a standard feedforward neural network with M layers and hidden size H. We built

[6] In retrospect, this was perhaps a naive hypothesis, given that the Hindi-English tweets are in a romanized alphabet, whereas the Hindi used for pre-training `bert-base-multilingual-cased` was presumably written mostly in Devanagari script.

Table 2: Experimental results for our bag-of-words models, varying the frequency threshold K. These experiments each use a two-layer network with a hidden size $H = 784$.

Frequency Threshold (K)	Accuracy
10	58.8%
15	58.6%
20	58.3%

Table 3: Experimental results for our bag-of-words models, varying the number of layers of the feedforward neural classifier. These experiments each use the same frequency threshold $K = 15$.

Number of NN Layers	Hidden Layer Size (H)	Accuracy
2	300	59.3%
2	768	58.6%
3	768	58.0%
4	768	57.8%

Table 4: Experimental results for our bag-of-ngrams models using unigrams, bigrams, and trigrams, using a frequency threshold $K = 15$ for all ngrams. These experiments use 2-layer feedforward neural networks with a hidden layer size $H = 300$.

Ngrams Used	**Accuracy**
uni	59.3%
uni + bi	60.0%
uni + bi + tri	60.0%

and trained the NNs using PyTorch (Paszke et al., 2019). In order to classify tweets into positive, negative and neutral, the final layer applies the softmax function to the 3-dimensional output of the NN, which normalizes the output into a probability distribution over the three possible sentiments of the input tweet. We used a cross-entropy loss function for training.

5.1 Variant: Count-of-Words

We also experimented with a count-of-words representation, where instead of just representing whether a vocabulary word appears in a particular tweet as a binary value, we instead represent the *frequency* of that word in the tweet. The motivation was that the classifier might gain insight into the sentiment of the overall sentence based on the frequency of 'positive' or 'negative' vocabulary members.

5.2 Variant: Bag-of-Ngrams

We also experimented with a "bag-of-ngrams" approach that extended our vocabulary by including ngrams up to length N (again using a minimum frequency threshold of K). Through this extension, we hoped the trained systems could exploit contextual information from neighboring words that are more than the sum of their parts, like "pretty good" or "awfully well done."

5.3 Experiments

For our experiments, we varied 3 hyperparameters: the frequency threshold K for the words/ngrams (Table 2), as well as the number of NN layers and hidden layer size H(Table 3).

Unsurprisingly, system accuracy improved (Table 2) as we decreased the frequency threshold K. This at least confirmed the natural hypothesis that a larger vocabulary size would lead to more information and improved sentiment analysis.

Increasing the number of layers was somewhat detrimental to model performance, as was widening the hidden layer size (Table 3). Perhaps this was caused by overfitting and might have been fixed by a regularizer, but nevertheless we stuck to simple two-layer networks for the remainder of the experiments.

After this first round of experiments, we conducted additional experiments using the count-of-words approach. In general, the accuracies in the count-of-words approach were slightly lower than the simple BoW approach.

For our bag-of-ngrams experiments (Table 4), the inclusion of bigrams improved the system results, while the addition of trigrams did not have much impact, likely because of the absence of sufficiently frequent trigrams (around 10, using a frequency threshold $K = 15$).

5.4 Takeaways

With only minor hillclimbing effort, we were able to train a simple bag-of-bigrams classifier to a validation accuracy of 60%. Contrast this to the validation accuracy of 62.2% achieved via fine-tuning of `bert-base-cased`, and the validation accuracy of 61.0% achieved via fine-tuning of `bert-base-chinese`. The similarity of these results suggests that the success of the BERT models is probably not due to a deep understanding of the code-mixed language, but rather the Transformer's ability to exploit simple word count statistics slightly better than a simple bag-of-words classifier.

6 Final System Submissions

Because we obtained slightly better performance with the fine-tuned BERT model, we used this as the basis for our competition system. To improve its performance, we experimented[7] with various hyperparameter settings, including *learning rate*, *weight decay*, *max_grad_norm* and *Adam epsilon*. Our best validation result of 63.8% was yielded by *bert-large-cased* using a learning rate of 2×10^{-5}, a weight decay of 0, a *max_grad_norm* of 1, and an Adam epsilon parameter of 1×10^{-8}. This system became our first submitted system, achieving an F-score of 69.9% according to the official scoring script.

For our second submission, we used bagging (Breiman, 1996) to make our BERT classifier more robust. We created 10 bootstrap samples from the given training data. We then trained 10 instances of the `bert-large-cased` model on these bootstrap samples, and then combined the results of all the models by voting, i.e., if a plurality of the 10 models predicted a particular tweet as positive (negative/neutral), then the tweets label is deemed as positive (negative/neutral). Our bagged system obtained an F-score of 71.3%, placing us 4^{th} overall in the competition out of 62 entrants.

7 Conclusion

In this paper we investigated sentiment analysis on code-mixed Hinglish (Hindi-English) tweets as participants of Task 9 of the SemEval 2020 competition. We implemented two main approaches: 1) applying transfer learning by fine-tuning pre-trained models like BERT and 2) training feedforward neural networks on bag-of-words representations. We found that the results of fine-tuning `bert-large-cased` (24 layers) could be approximated by a 2-layer BoW feedforward NN. During the evaluation phase of the competition, our top system obtained an F-score of 71.3%, placing 4^{th} out of 62 entries in the official system rankings.

The fact that we managed fourth place with a system that was little better than a bag-of-words classifier suggests that existing pre-trained vector representations are not particularly good models of code-mixed language. One obvious reason is the difference between the pre-training domain (English Wikipedia) and the target domain (code-mixed tweets). While we were initially hopeful that the multilingual BERT model released by Google might be more effective than the English-only models, it was not. It remains unresolved whether this is because of the formality differences between the two domains, because of the lack of romanized Hindi in Google's training data, or because language models simultaneously pre-trained on multiple languages do not generalize sufficiently to properly understand code-mixed data.

[7] We also experimented with the *roberta-base* model (Liu et al., 2019), but were not able to improve of BERT's results.

References

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24(2):123–140.

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*.

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

Michael Frederick McTear, Zoraida Callejas, and David Griol. 2016. *The conversational interface*, volume 6. Springer.

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 *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc.

Parth Patwa, Gustavo Aguilar, Sudipta Kar, Suraj Pandey, Srinivas PYKL, Björn Gambäck, Tanmoy Chakraborty, Thamar Solorio, and Amitava Das. 2020. Semeval-2020 task 9: Overview of sentiment analysis of code-mixed tweets. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, December. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.