Team_Swift at SemEval-2020 Task 9: Tiny Data Specialists Through Domain-Specific Pre-training On Code-Mixed Data

Aditya Malte
Department of Computer Engineering,
Pune Institute of Computer Technology,
Pune, India
aditya.malte@gmail.com

Pratik Bhavsar
TaskHuman
Mumbai, India
pratik.a.bhavsar@gmail.com

Sushant Rathi
Department of Computer Science and Engineering
Indian Institute of Technology,
Delhi, India
rathisushant5@gmail.com

Abstract

Code-mixing is an interesting phenomenon where the speaker switches between two or more languages in the same text. In this paper, we describe an unconventional approach to tackling the SentiMix Hindi-English challenge (uid: aditya_malte). Instead of directly fine-tuning large contemporary Transformer models, we train our own domain-specific embeddings and use them for downstream tasks. We also discuss how this technique provides comparable performance while making for a much more deployable and lightweight model. It should be noted that we have achieved the stated results without using any ensembling techniques, thus respecting a paradigm of efficient and production-ready NLP. All relevant source code shall be made publicly available to encourage the usage and reproduction of the results.

1 Introduction

With the rapid growth in the number of active Internet users, there comes a commensurate need to analyze their usage patterns. There may be several reasons to do so, including content moderation, cyber-abuse detection and even for market research purposes to enhance the user experience.
However, considering the petabytes of data available online, it becomes a mammoth task that is nearly unachievable through manual means. For instance, social media platforms like Twitter have millions of tweets generated by users every day. By abstracting away tasks like sentiment analysis and content moderation, Natural language processing (NLP) techniques can play an instrumental role in this kind of data analysis.
SentiMix Hindi-English (Patwa et al., 2020) is a code-mixed sentiment analysis task. Code-mixing is a phenomenon where the author arbitrarily switches between two or more languages in the same snippet of text. Code-mixing, along with the inherently noisy nature of social media text make its analysis a non-trivial task. We experimented with a wide range of techniques that included contemporary Transformer architectures (Vaswani et al., 2017), word embedding pre-training and language model pretraining. Our experiments provide an interesting inference that near state-of-the-art performance need not require massive model sizes. This analysis would especially be helpful for those trying to deploy machine learning models in production, where computational power and inference speed may be a constraint.

2 Background

In this section, we provide a short discussion of the algorithms and technologies that are a part of our implementations.

This work is licensed under a Creative Commons Attribution 4.0 International Licence. Licence details: http://creativecommons.org/licenses/by/4.0/.

Proceedings of the 14th International Workshop on Semantic Evaluation, pages 1310–1315
Barcelona, Spain (Online), December 12, 2020.

2.1 Distributed Word Representations

Distributed Word Representations are fixed-sized vectors that abstractly represent the semantics of a word. Word representations rapidly gained popularity and have become an important tool in the NLP toolbox. An example of word embedding includes the word2vec (Mikolov et al., 2013) model that can either be trained using the skip-gram (where surrounding words predict a context word) or the CBOW (where the context word is used to predict surrounding words) algorithm. However, an important drawback of these word2vec is its reliance on a word: word-embedding mapping. Not only does a one-to-one mapping like this increase the size of vocabulary, but it also makes the model more susceptible to spelling errors and variations.

2.2 Code-Mixed Sub-Word Representations

To overcome such problems, we focus on extracting features at a sub-word level. In this paper, we shall focus our discussion on $FastText$ representation. $FastText$ (Bojanowski et al., 2016) is a sub-word representation that uses a sliding window approach to generate subword embeddings subsequently used to calculate a word embedding.

Code-mixed Hindi-English tweets have a high probability of suffering from both of these issues. For example, the Roman transliteration of "हमारा" (meaning "ours" in Hindi) is valid even if it is transliterated to "humara", "humaara", "hmaara" or their combinations. Furthermore, we observed that the transliteration of social media text from Hinglish to Hindi is inconsistent for noisy text.

Thus, FastText is an effective tool in addressing such shortcomings of traditional word-based models.

2.3 Bi-LSTM

LSTM (Hochreiter and Schmidhuber, 1997) networks follow a recurrent network architecture, where the tokens of a sequence are fed into LSTM *cells*. These are memory cells which help the model capture long-range dependencies in data, giving vastly improved results on sequence classification over traditional RNNs.

Bidirectional LSTM networks (Bi-LSTM) augment the LSTM to make use of both past and future tokens. This 'forward' context provided helps the model avoid bias towards the starting tokens of a sentence, and Bi-LSTM is now used as a standard baseline for any sequence classification task.

2.4 XLM-RoBERTa

The Transformer architecture has given phenomenal results in NLP when compared to traditional RNN-based techniques. These were further improved through the BERT (Devlin et al., 2018) model that introduced bidirectionality in the self-attention mechanism. Consequently, the RoBERTa (Liu et al., 2019) model gave us insights as to how the BERT can be improved by training longer and with more data. Further changes by RoBERTa included the elimination of the Next Sentence Prediction pre-training task and dynamic masking. XLM (Lample and Conneau, 2019a) introduced a new LM task Translation Language Modeling (TLM) to help learn cross-lingual embeddings.

Alongside combining the benefits of the RoBERTa and XLM model with longer pre-training on more data, XLM-RoBERTa (Lample and Conneau, 2019b) has the added advantage of having Romanized Hindi as part of its pre-training set.This inductive bias could help for better adaptation for task data.

3 Dataset Description

The organizers have provided a Hindi/English code-mixed sentiment analysis dataset consisting of samples labeled as negative, neutral and positive. The dataset was provided with language labels per token (en-English, hi-Hindi, O-universal). The distribution is as shown in Table 1.

The dataset consists of informal code-mixed text with the widespread use of slang words and colloquial references. This, along with some label noise makes training the model an especially interesting task. The dataset also contains emojis. The above-mentioned attributes make for a dataset that reliably represents the realistic scenario of analyzing noisy Twitter data.

Category	Count
Positive	4634
Neutral	5264
Negative	4102
Total	14000

Table 1: Distribution of training data

4 Pre-training/Fine-tuning Corpus Generation

While FastText still provides an embedding for OOV tokens from the Hinglish text, the embeddings so generated would be of suboptimal quality due to a large number of unseen tokens of the Hinglish text. To address this problem, a Hinglish corpus is generated by scraping publicly available tweets from Twitter. This corpus is also used in our alternate approach of fine-tuning the XLM-R language model to be subsequently used for classification. Additional Efforts are made to generate a corpus that covers domains relevant to the SentiMix dataset by utilizing features such as popular user IDs in the training sets. A small minority of tweets present in pure Devanagiri script were removed to retain only Hinglish tweets in Roman script. Thus, we generated a corpus consisting of approximately 5.7 million tokens (≈ 306259 tweets).

4.1 Text Preprocessing

General text-preprocessing approaches such as the following have been used:

- Lowercasing
- Removal of repeated punctuations
- Removal of numbers and URLs

Interestingly, we chose to retain emojis rather than converting them to their textual equivalents (also known as demojization). Furthermore, user tags (@), have been retained as they could be an essential feature in detecting that the tweet is targeted at an entity.

5 Proposed Approach

For the said task, we attempted multiple techniques and their combinations. In this paper, we have limited our discussion to the XLM-R and the HiText Bi-LSTM related approaches.

5.1 XLM-RoBERTa

Bidirectional Transformer architectures have been previously used for code-mixed text classification. (Malte and Ratadiya, 2019) have utilized multilingual BERT (mBERT) for the code-mixed hate speech classification task. We utilize the XLM-RoBERTa model for this text classification task to leverage its pretraining on Romanized Hindi text (unlike the mBERT model). Since public models are trained on generic data, we decided to follow a domain adaptation approach as Hinglish tweet data is different from the training data of XLM-R. Firstly, we performed the LM-finetuning on the domain-specific corpus that we had scraped from Twitter. As shown in Table 2, LM fine-tuning has provided significant improvement. We then train the model with classification head and cosine warmup for several epochs. Experimentally, we found the following hyperparameters provided good performance:

- Batch size - 64
- Label smoothing - 0.2
- Learning rate - $8e{-5}$ to $4e{-5}$
- Sequence length - 192

As demonstrated in Table 2, it is worth noting that LM Finetuning and label smoothing provided noticeable improvements over direct classifier fine-tuning. We have illustrated the confusion matrix for the same in Fig. 2.

5.2 Bi-LSTM + HiText

A FastText model of latent layer size 200 was trained from scratch on the generated Hinglish Twitter corpus. Our experiments showed that we were able to obtain high-quality word/token representation that captured sentiment, context, grammatical clues and also semantic understanding of nouns. Our choice of not demojizing the text allowed a common hyperspace of emojis and plain text. This translated to interesting and useful emoji-emoji representation proximity such as 🙏 - 😊 along with emoji-word representation proximity like 🙏 - "pranam" (a respectful greeting).

The trained Hinglish FastText embedding is then passed to a Bi-LSTM neural network for performing sequence classification.

We experimented with the number of hidden layers (n), size of the hidden layer (s), and weight decay (w) as well as dropout (p) for model regularization. Best results on dev set were found for n = 3, s = 256, w = 0.0005 and p = 0.15. The Adam optimizer was used for training the model. Weight decay was experimented with in order to overcome co-related features in the HiText embedding and avoid overfitting.

5.2.1 Rule-Based Feature Extraction

Certain rule-based features such as the count of emojis and punctuation could give important clues relating to the sentiment and polarity of a sentence. For instance, the presence of several exclamation marks (!) may indicate extreme polarity in either direction (positive or negative). Similarly, the ratio of Hindi/English tokens in a given text could help predict the emotion of a text. Furthermore, a large count of capitalized text may indicate aggression in a tweet. We have thus extracted the following features from the text:

- Count of English, Hindi and Universal tokens
- Ratio of English to Hindi tokens
- Count of punctuation
- Number of Capital letters in the text

Classification is performed using the HiText-BiLSTM, rule-based features and the confusion matrix for the same is illustrated in Fig. 1.

6 Results

Model	F1
Vanilla XLM-R	0.652
XLM-R + LM-Finetuning + Label Smoothing	0.69
HiText-BiLSTM	0.677
HiText-BiLSTM+Rule-based Features	0.69
XLM-R + LM-Finetuning + Label Smoothing + Rule-based Features	0.69

Table 2: Performance on Test Dataset

Model	Time (s)
XLM-R	65.15
HiText-BiLSTM	3.92

Table 3: Inference Speed

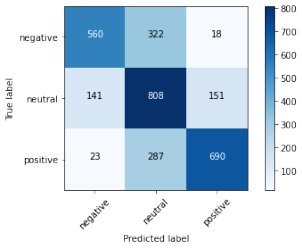

Figure 1: HiText Bi-LSTM + Rule-based

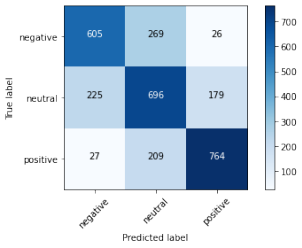

Figure 2: LM-tuned XLM-R

The results of our experiments have been described in Table 2 and Table 3. Inference speeds were recorded on a Tesla K80 GPU on a sample size of 3000. Results for *Vanilla XLM-R* and *HiText-BiLSTM+Rule-based Features* are available on the competition website itself, while the other results were calculated after test labels were released by the organizers. Our highest scoring submission *HiText-BiLSTM+Rule-based Features* reported class-wise F1 scores of 0.69, 0.64, 0.74 on the classes negative, neutral and positive respectively, ranking 10^{th} on the final leaderboard. While one would generally expect the XLM-R model to far outpace the HiText Bi-LSTM model, our results show that domain-specific pre-training allows for comparable performance, while still allowing deployment in a resource-constrained environment

7 Discussion

Our experimental results demonstrate that using task-specific (Hinglish tweets in this case) training data helps models perform better compared to their generic alternatives. We achieved a noticeable jump in the performance of the XLM-R thanks to fine-tuning the language model on a task-specific corpus.

However, as part of its pre-training set, the XLM-RoBERTa model is trained on only 0.5 GiB of general-purpose Romanized Hindi, and is thus not trained specifically on noisy social media data. Thus, it is observed that a much lighter and simpler model such as the Bi-LSTM performs comparably well when coupled with our HiText model that is trained exclusively on domain-specific data. This leads us to an interesting observation that very small models trained on very small amounts of domain-specific data may provide performance that is comparable to huge Transformer models trained on generic corpora.

As is visible from Table 3, our BiLSTM model provides significantly higher inference speed when compared to the XLM-R model. This further reinforces our statement that domain-specific models may be better suited in the production environment, even if the said domain-specific corpus is significantly smaller in size.

We also find that the performance of the XLM-R model is drastically improved upon domain-specific training. The addition of label smoothing also allows the model to better handle a minority of noisy or mislabelled samples in the dataset. However, the addition of rule-based features does not benefit the XLM-R model as it did with the HiText Bi-LSTM one.

Importantly, the absence of any model ensembling techniques makes these models more explainable and faster in a production environment.

8 Conclusion

Through this paper, we have demonstrated how task-specific pre-training of very small models can give a performance that is comparable to classifier fine-tuning on massive generic publicly available models. One should also realize the advantages that small models would have concerning infrastructure costs and scalability when deployed in production. With inference speed nearly twenty times using such small scale models, we have shown how simple models can provide good deployability without compromising on performance.

All relevant source code concerning dataset generation, preprocessing and model training has been made publicly available for the usage and reproduction of our results.[1]

[1] https://github.com/aditya-malte/SemEval-SentiMix

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *CoRR*, abs/1607.04606.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

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

Guillaume Lample and Alexis Conneau. 2019a. Cross-lingual language model pretraining. In *NeurIPS*.

Guillaume Lample and Alexis Conneau. 2019b. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

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*, abs/1907.11692.

A. Malte and P. Ratadiya. 2019. Multilingual cyber abuse detection using advanced transformer architecture. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 784–789.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints*, page arXiv:1301.3781, Jan.

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.

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