Sentimiento Analysis: Classifying Affect in Mixed-Language Tweets

Connor Boyle boyle128@uw.edu
Martin Horst mmhorst@uw.edu
Nikitas Tampakis tampakis@uw.edu

1 Introduction

Although a large percentage of the world's population is multilingual, most natural language processing (NLP) tasks focus on monolingual contexts. Our project aims to investigate how state-of-the-art machine learning approaches to sentiment analysis perform on code-mixed (i.e. multilingual) sentences. Speakers of Spanglish and Hinglish (Spanish-English code mixing and Hindi-English code-mixing, respectively) naturally blend lexica and syntax across languages. Can language models from each language be combined to correctly interpret affect in tweets? Do features like emoji, punctuation, and orthographic variation also play a role in the emotion expressed in a tweet? We demonstrate that although accuracy of sentiment classification can be improved by leveraging state-of-the-art methods, there is still substantial improvement to be made in this NLP task.

2 Task Description

Our primary task consists of classifying code-mixed Spanish-English social media (Twitter) posts with sentiment labels, which was part of SemEval 2020's Task #9: SentiMix (Patwa et al., 2020). Our adaptation task is the analogous Hindi-English subtask of the same SemEval 2020 task.

The training and testing data sets of each language sub-task are both comprised of tokenized mixed-language tweets, with token-level language labels and sequence-level (i.e. tweet-level) sentiment labels. The sentiment labels are each one of three categories: "positive," "neutral," or "negative."

Submitted classifier models were evaluated based on their multi-class support-weighted F1 score on held-out evaluation ("test") data.

2.1 Our Data

The fully-labeled Hinglish training and test data (20,000 tweets), as well as fully-labeled Spanglish training (but no test) data (18,789 tweets) are publicly available (on the competition website). The same organizers of the original shared task later opened a new task (with its own train, dev, and test split) on the LinCE evaluation platform (Aguilar et al., 2020). We used the "train-dev-test" data split (whose instances overlapped significantly, but not entirely, with the original task), for the final evaluation stage of our last deliverable.

For the purposes of this paper, the "train-dev-test" dataset from the original SemEval 2020 paper will be referred to as the "Spanglish Sentimix" dataset, and the "train-dev-test" dataset from the LinCE platform will be referred to as the "Spanglish LinCE" dataset.

2.1.1 Language Tags

The shared task organizers provided language tags for each of the tweet tokens. The Spanglish data was manually tagged, while the Hinglish data was machine-tagged.

Tag	Description
lang1	English word
lang2	Spanish word
ne	named entity
ambiguous	Spanish or English word (i.e. "no")
unk	tagger unable to identify word
other	punctuation, url, emojis
mixed	mixed Spanish/English word
fw	foreign language word
Eng	English word
Hing	Hindi word
O	Everything else

Table 1: Spanglish followed by Hinglish (last 3) Tags

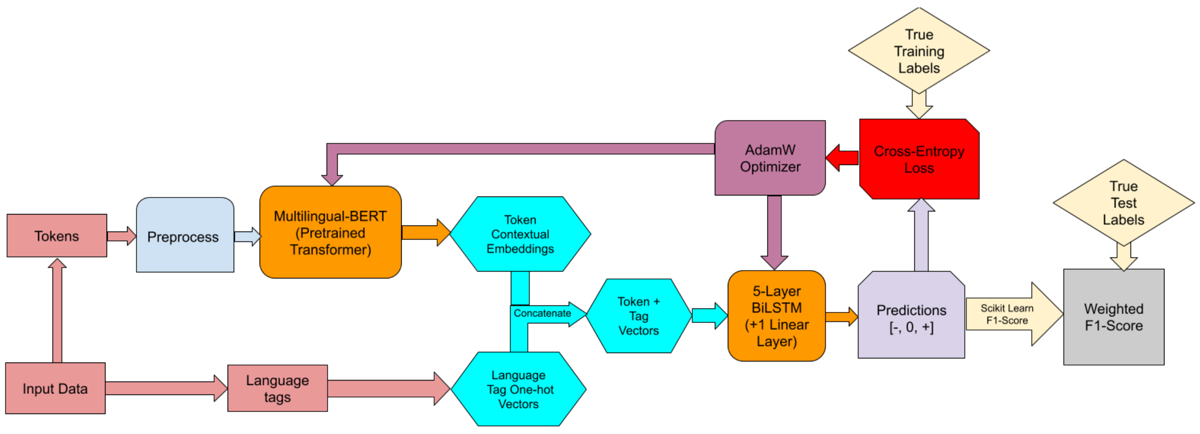

Figure 1: System architecture diagram for D4

3 System Overview

3.1 D2

In the first iteration of our affect recognition system (D2), we trained linear SVMs using a bag-of-n-grams ($n \in \{1, 2, 3\}$) representation of the input tweets and language tags (i.e. separate n-grams of tokens and n-grams of tags). We used the tokenization exactly as was provided. This system achieved a weighted F1 score of 44.9% on the Spanglish "dev" dataset, and 57.9% on the Hinglish "dev" dataset.

3.2 D3

For our next system architecture (D3) we fine-tuned an instance of the Bidirectional Encoder Representations from Transformers (BERT) model presented in Devlin et al. (2018). Our model is based on the `bert-base-multilingual-cased` ("M-BERT") checkpoint, which was trained on a masked-language modeling (MLM, i.e. Cloze testing) task on content in over 100 different languages. It can be adapted to sequence classification by replacing the MLM output model with a single classification layer. This deliverable wholly excluded the language tags from the vectorized representations of the code-mixed tweets.

3.3 D4

For our last system architecture (D4, shown in Figure 1) we added a preprocessing step to normalize URLs (regardless of original tokenization), twitter mentions (i.e. "@handle_name"s) and translate emoji (which were generally out-of-vocabulary for BERT's tokenizer) into English words. We used BERT to convert this preprocessed text into sequences of contextual token embeddings; we then concatenated one-hot vectors representing language tags to these contextual embeddings and and fed them into a deep Bidirectional Long Short-Term Memory (BiLSTM) classifier.

4 Approach

4.1 Preprocessing

The original Hinglish data was tokenized in a slightly different way from the Spanglish tweets. Whereas in our Spanglish data, URLs and retweets were kept as one token each, the URLs and mentions in Hinglish tweets were split into multiple tokens. For example, a given URL would be presented as `['https//t.co/quG4aCRVx8']` in Spanglish, but as `['https', '//', 't', '.', 'co', '/', '.', 'quG4aCRVx8']` in Hinglish. To account for this, we wrote rules to recombine these artifacts, and then replace them with the strings "HTTPURL" and "@USER" as per Chiorrini et al. (2021).

The latest preprocessing step we took was to convert emojis into text representation using the `emoji_translate` Python package.

These tokens were fed into the pre-trained BERT wordpiece tokenizer for `bert-base-multilingual-cased` from the Transformers library (Wolf et al., 2020). The model token vocabulary segmented tokens that were either entire words, single characters, or frequently-observed subword components. Based on the resulting F1 scores, we observed a larger improvement due to preprocessing on the Hinglish data sets than the Spanglish data sets, most likely due to the difference in original tokenizations.

4.2 Model training

We trained our model using the AdamW optimizer with a learning rate $\eta = 2e^{-5}$ and epsilon $\epsilon = 1e^{-8}$, as in the baseline models for the shared task (Patwa et al., 2020). We used a linear learning rate scheduler, which scaled down to zero after all training steps (i.e. the number of epochs times the per-epoch number of batches).

We followed a tutorial by (McCormick and Ryan, 2019) in order to produce a fine-tuned model capable of classifying our code-mixed tweets. The principle differences for our system were our base model (we used the `bert-base-multilingual-cased` model checkpoint) and the training data (sentiment of code-mixed tweets, instead of grammaticality judgements on full English sentences). We used Google Colab to train our model, whose free computational resources were a boon to our development process. For example, a model trained using a (free) Colab GPU for 4 epochs finished training in under 20 minutes, compared to an estimated 6 hours on a local, high-end CPU.

4.3 Ablations

Frozen Layers	Dropout	Avg F1	Best Epoch
-1	0.0	0.6014	3
1	0.0	0.5785	2
3	0.0	0.5886	4
-1	0.5	0.5426	2
1	0.5	0.5736	3
3	0.5	0.5688	3
-1	0.75	0.5378	4
1	0.75	0.4625	9
3	0.75	0.4551	4

Table 2: Experimenting with introducing dropout and freezing layers of BERT on the Hinglish dev data. The first entry, which represents a model with no frozen layers and no dropout introduced, performed the best.

After recording results with our final classifier architecture (the version of our code that was tagged "D4"), we suspected that catastrophic forgetting or some other form of overfitting were occurring during the training process. Therefore, we considered ablations such as freezing the bottom n layers of BERT (plus the global embedding layer) as well as adding dropout in our LSTM classifier layer. We tried freezing up to a quarter of the bottom layers, as recommended by Lee et al. (2019). Since these ablated versions could potentially benefit from more training, we changed the number of training epochs to 10 for our ablation runs, and retrained the model using the Hinglish training data (with the "dev" dataset for evaluation). None of the ablations ultimately led to a higher-performing model.

5 Results

See Table 4 on the next page for in-depth results on both the primary and adaptation tasks. Evaluations on the "dev" file of the original Spanglish Sentimix dataset are referred to as Spanglish "Spanglish Dev"; evaluations on the "test" file of the Spanglish LinCE dataset (Aguilar et al., 2020) are referred to as "Spanglish Eval". The results of our "Spanglish Eval" can be compared with the following other submissions on the LinCE site as shown in table 3.

System	Avg. F1
M-BERT+BiLSTMClassifier	0.4849
Baseline (M-BERT)	0.5643
Top Performer	0.6126

Table 3: Comparing our results in row 1 with the weighted F1 scores provided on the LinCE submission leaderboard.

6 Discussion

Our final system performed measurably worse when evaluated on the "dev" Spanglish set (Spanglish Sentimix) than our less complex D3 model (simple fine-tuned BERT) on the same dataset. When evaluated on the held-out test data, our D4 system also fell well short of the Spanglish baseline desc in the overview paper (which was evaluated on the original Spanglish Sentimix test dataset, whose labels are no longer accessible), as well as the baseline listed on the new Spanglish LinCE competition website. Both of these baselines were trained by fine-tuning multilingual BERT.

Our final system maintained similar performance on the Hinglish "dev" dataset, as compared to D3. Since our D3 model was a direct copy of the shared task baseline, we did not expect our D4 implementation to exceed the baseline performance on the final eval set. However, our final model *did* surpass

System	Positive			Neutral			Negative			Avg F1
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	
Spanglish Linear SVM (D2)	0.5386	0.7499	0.6252	0.3689	0.2628	0.2971	0.4157	0.1577	0.2233	0.4486
Hinglish Linear SVM (D2)	0.6500	0.6448	0.6432	0.5372	0.4505	0.4838	0.5841	0.6930	0.6302	0.5794
M-BERT - Spanglish (D3)	0.5981	0.6348	0.6159	0.4028	0.4296	0.4158	0.4253	0.2925	0.3466	0.5041
M-BERT - Hinglish (D3)	0.6568	0.6782	0.6673	0.5401	0.4832	0.5101	0.6059	0.6652	0.6342	0.5984
Spanglish Dev (D4)*	0.5767	0.7576	0.6549	0.3791	0.2052	0.2663	0.3963	0.3854	0.3908	0.4815
Spanglish Eval (D4)	0.6288	0.8854	0.7354	0.1	0.0008	0.0015	0.3826	0.4784	0.4252	0.4849
Hinglish Dev (D4)*	0.6790	0.6721	0.6755	0.5311	0.5000	0.5151	0.6025	0.6539	0.6272	0.6008
Task Baseline Hinglish	0.728	0.688	0.707	0.562	0.602	0.581	0.691	0.674	0.683	0.654
Hinglish Eval (D4)	0.7774	0.7370	0.7567	0.6013	0.5936	0.5974	0.6894	0.7400	0.7138	0.6854
Top Hinglish Eval Score	0.843	0.760	0.799	0.652	0.731	0.689	0.785	0.754	0.769	0.750

Table 4: Results across the different system approaches for both the primary and adaptation task.
(*) Note: An incorrect version of the model was submitted at the D4 checkpoint. The correct version is provided here and submitted separately with our other models on Google Drive.

the baseline for Hinglish by 3 percentage points (from an F1 of 0.654 to 0.685), when evaluated on the final, held-out "test" data file.

6.1 Imbalanced Class Splits

The Spanglish and Hinglish datasets differed significantly in distribution of positive, neutral, & negative labels in the dataset. While Hinglish had an equal distribution across the three classification labels, Spanglish had a more skewed distribution. In the Spanglish LinCE dataset that was used for the final evaluation of our model, 55% of the tweets were labeled positive, 30% neutral, and 15% negative. This distribution was present in all three sections of the data (train-dev-test). The higher occurrence of positive labels caused our model to more frequently classify the test instances as positive and not classify many instances as neutral. The neutral label had an abysmal 0.0015 F1 score when run on the LinCE dataset. A more deliberate effort to resample the dataset to provide more balanced training examples may have led to a resulting model that could have better differentiated across the three classes in the Spanglish eval task.

7 Conclusion

We learned over the course of this project that even relatively easy-to-use off-the-shelf neural models such as BERT can still be fairly difficult to train, and that the number of variations in regularization techniques other training configurations make it difficult to replicate a particular model architecture from a paragraph-long description. We *believe* that our D3 system was identical to the system in the overview paper, however we were not able to achieve comparable performance with the baseline. This was particularly true with Spanglish, where there was a discrepancy of about 15 percentage points from the baseline model to our own. Changing hyperparameters of token length, preprocessing the input data, and swapping out data sets were approaches that yielded no significant performance gains, and without knowing exactly how the baseline system was trained we are left marginally in the dark. We can only assume that with more practice and patience we can master the art of training (and debugging) neural language models.

While we are pleased to have nominally beaten the SentiMix overview paper's baseline for Hinglish, the fact that we did not see similar improvement in Spanglish leads us to believe that these gains may not necessarily be meaningful or generalized.

Ultimately, our much more sophisticated neural model(s) did not do dramatically better than the significantly simpler shallow model we trained in our first iteration. This was particularly evident in the D2 versus D3 Hinglish data set, where our F1 score for D3 jumped only 2 percentage points despite having tens of millions more parameters than the previous system. The trend in NLP at the moment is to maximize training time, number of parameters, or computational power, so it was eye opening to find that a simple bag-of-n-grams model —which was trainable in minutes on a CPU-only desktop —could perform nearly as well. Perhaps even more importantly, the earlier model requires far less storage space and requires far less computation at inference time, thus making it far more practical for, say, client-side use.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. Lince: A centralized benchmark for linguistic code-switching evaluation.

Andrea Chiorrini, Claudia Diamantini, Alex Mircoli, and Domenico Potena. 2021. Emotion and sentiment analysis of tweets using bert. In *EDBT/ICDT Workshops*.

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*.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *CoRR*, abs/1911.03090.

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. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.