SemEval 2022 Task 4 Subtask 1 Binary Classification of PCL

Srinivas Akhil Mallela
srma3452@colorado.edu

Aravind Bisegowda Srinivas
arbi8560@colorado.edu

Abstract

In this paper we discuss the results we obtained while attempting the SemEval 2022 Task 4 Subtask 1 of identifying whether patronizing and condescending language is present in a given body of text. We train our language models on the 'Don't Patronize Me!' dataset provided by Perez-Almendros et.al. After having tried various approaches, transfer learning via the XLNet pre-trained model gave us the best results with scores of **0.43** precision, **0.78** recall and **0.56** F1.

Introduction

Condescending language and patronizing language (PCL) can bring dialogues to an end and divide communities. Detecting such language is an impactful task in the field of linguistics that could open the door to many applications and future research directions. A major hurdle to develop models to detect PCL in texts is the lack of a high-quality labeled dataset. A broader limitation is that PCL is often impossible to detect from isolated utterances. PCL is not overtly negative or critical. Sometimes it might even include praise. Furthermore, condescension tends to rest on a pair of conflicting pragmatic presuppositions. For example, an utterance that is entirely friendly if said by one friend to another might be perceived as highly condescending if said by a customer to a store clerk. In such cases, the social roles of the participants shape the language in particular ways to yield two very different outcomes.

The aim of this task is to identify PCL, and to categorize the linguistic techniques used to express it, specifically when referring to communities identified as being vulnerable to unfair treatment in the media. We use the 'Don't Patronize Me' introduced by Perez-Almendros et.al. to train our language models. We have tried approaches using LSTM, Transfer learning and there is detailed information about the implementations further along in this report.

Background

The ‘Don’t Patronize Me!’ (DPM) data set created by Perez-Almendros et.al. currently contains 10,637 paragraphs about potentially vulnerable social groups. These paragraphs have been selected from general news stories and have been annotated with labels that indicate the type of PCL language that is present, if any. The paragraphs have been extracted from the News on Web (NoW) corpus2 Davies, 2013.

Preprocessing was done with standard techniques such as:

- **Removal of stopwords**
  Stopwords are essentially the most common words in any language that do not add much information to the text. We use the nltk stopwords library to remove stopwords from paragraphs of the DPM dataset.

- **Stemming/lemmatization**
  The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. We use the nltk library’s PorterStemmer and WordNet Lemmatizer to stem and lemmatize the DPM paragraphs.

- **Tokenization**
  Tokenization is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. We use the nltk.word_tokenize library to tokenize the paragraphs in the DPM dataset.

Transfer learning seems to be the most intuitive approach to solve this language classification task and we experimented with the following three pre-trained models: BERT, RoBERTa & XLnet. We tried a bidirectional LSTM approach as well to compare its metrics with transfer learning.

System Overview

First, we implemented Bi-Directional LSTMs as they are traditionally used for classification problems. Multi-layered Bi-Directional LSTMs each with 100 units representing dimensionality of the outer space, and input embedding size of 100 and max sequence length of 250 were used.

Preprocessing was done with standard techniques such as lemmatization, removing stopwords & vectorization is a standard dictionary - unique value approach. Early stopping factors on validation (10% split) loss and hyper parameters tuning helped achieve a reasonable F-1 score on a randomized and shuffled 80-10-10 train-dev-test split. Dropout and recurrent dropout of 0.10 was used to help prevent overfitting. An optimal threshold for prediction was also computed via precision-recall curve to counter the imbalanced training set.

We moved to transfer learning with BERT (Bert-base-uncased), a pre-trained bidirectional transformer model. BERT results show that it can contextualize a deeper sense of language flow and learning when compared to traditional models. Using the BERT tokenizer and encoder, the train text data is passed to the BERT layers and the output is passed through a dropout layer of 0.1 to prevent overfitting and a final sigmoid activation function. After training for 10 epochs with a batch size of 32, the results were better than LSTM. Downsampling imbalanced data is a significant contributor to improving the performance and a split of 1:2.5 for the labels helped improve the scores significantly for a randomized and shuffled 80-10-10 train-dev-test split. The default threshold for interpreting probabilities to binary class labels is 0.5, but since we have a skewed training distribution, calculating the optimal threshold for prediction via precision-recall curve improved the scores further.

Finally, transfer learning with XLNet using the SimpleTransformers framework produced the best scores both on a randomized 80-10-10 train-dev-test and also the organizers' split. XLNet is a generalized autoregressive pretrained method and it is the kind of model that uses the context word to predict the next word. We are using the tokenizer and vectorization already provided by the SimpleTransformers framework. Establishing weights of 0.25 and 6 to labels while downsampling the training data (1:2.5 split) helped utilize the skewed training data well. The whole purpose is to penalize the misclassification made by the minority class by setting a higher class weight and at the same time reducing the weight for the majority class. Hyper parameter tuning also helped specifically training batch size and early stopping based on Matthews correlation coefficient (Patience of 3 epochs) which is a reliable statistic. It produces a high score only if prediction produces good results in all four confusion matrix categories (false negatives, true positives, true negatives & false positives).

Experimental Setup

The original dataset is shuffled/randomized every time and split into 80-10-10 train/dev/test split. Seeds can be used to test particular splits but all the results mentioned below are on shuffled/randomized splits.

Here is a description of the language models we experimented transfer learning with:

BERT
BERT, which stands for Bidirectional Encoder Representations from Transformers is conceptually simple and empirically powerful language model. It is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. The pre-trained BERT model can be fine tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications.

RoBERTa
RoBERTa builds on BERT's language masking strategy. The system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. RoBERTa, which was implemented in PyTorch, modifies key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective, and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance.

XLNet
XLNet is a generalized autoregressive pretraining method that enables learning bidirectional contexts by maximizing the expected likelihood over all permutations of the factorization order and overcomes the limitations of BERT through its autoregressive formulation. XLNet integrates ideas from Transformer-XL, the state-of-the-art autoregressive model, into pretraining. Empirically, under comparable experiment settings, XLNet outperforms BERT on often by a large margin, including question answering, natural language inference, sentiment analysis, and document ranking.

Hyperparameter Tuning with early stopping, weight decay, learning rate, label weights & label downsampling helped in evaluating and improving the performance of all the three models we implemented.

Here are the key evaluations metrics we used to assess all our implementations:-

- **Accuracy** is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations.
- **Recall** is the ratio of correctly predicted positive observations to the all observations in actual class.
- **Precision** is the ratio of correctly predicted positive observations to the total predicted positive observations.
  F1 Score is the harmonic mean of precision and recall.
- **MCC** is arguably a good metric for binary classification problems. It takes into account true and false positives and negatives and is generally regarded as a balanced measure.

Here are the external libraries/tools used:

- Pandas
- Numpy
- NLTK
- Tensorflow
- Keras
- Simple Transformers
- TensorFlow Hub

Results

The F1 score obtained for the **transfer learning approach with the XLNet** model is **0.56** on the organizer's test-train split. At the time of this report's submission, this places us in the top 5 of the SemEval leaderboard.

The following scores were obtained with the randomized and shuffled 80-10-10 train-dev-test split of the 'Don't Patronize Me' dataset.

	F-1	**Precision**	**Recall**
LSTM	0.38	0.33	0.44
BERT (Tf-Hub)	0.48	0.35	0.73
XLNet	0.57	0.45	0.77

On organizers' split of train-test data, the XLNet model with a 1:2.5 downsampling of labels and label weights of 0.25 and 0.6 achieved **0.43 precision, 0.78 recall, 0.56 F1 score.** The other important hyperparameters were batch size of 32 and early stopping on MCC Metric.

Looking at our system's predictions, it's clear to see that the precision is consistently not doing well. Tweaking the optimal threshold for prediction or adjusting the weights for labels will improve the performance but it will be at the cost of other metrics as false positives, false negatives & true negative scores will definitely be affected.

Conclusion

Detecting patronizing and condescending language is definitely a difficult task and even with conceptually different approaches, it's been hard to capture PCL well. Improving a particular metric always brought down the score of other metrics making it hard to fine tune any of our approaches. Further experimenting with transfer learning on language models and more intricate layers and weights involved is a potential way to improve performance.