# Identify frames in the Tweets of US politicians

- Ganga Meghanath

***Abstract*—In this work, we utilize pre-trained sentence embeddings from Distilled RoBERTa to encode Congressional Tweets and classify them into 17 frames using a feed forward neural network. We use sentence similarity and data annotation on 3.2 million unlabelled tweets to further improve our predictions.**

## I. Framing

Framing refers to wording your opinion on a certain subject to emphasize certain aspects of the topic over the others. You can get insights into the views and standings of an individual with respect to an issue based on the words used by the individual and the way they frame their sentences. In real world scenarios, associating wordings and sentences from an individual or an organization to various frames can help us identify their bias with respect to the ongoing issue under consideration. This is a very powerful tool. For example, when it comes to elections, associating frames to candidates and political parties can help us correlate their views with that of ours, which could potentially have a huge influence on the votes that they receive. Another example is an organization's standing with respect to how environment friendly their operations need to be. Hence, frameworks for associating language to frames have numerous applications in the society.

## II. Description of Baseline

### A. Labelled Data

The labelled data comprises of 1230 tweets with the label distribution depicted in Fig. 1.

| Count of text | issue | | | | | | |
|---|---|---|---|---|---|---|---|
| Frame | abort | aca | guns | immig | isis | lgbt | Grand Total |
| (SELF) PROMOTION | 4 | 58 | 23 | 22 | 52 | 5 | 164 |
| CAPACITY & RESOURCES | 3 | 5 | 2 | 1 | 2 | | 13 |
| CRIME & PUNISHMENT | | | 2 | 1 | 5 | | 8 |
| CULTURAL IDENTITY | | | 4 | 8 | 6 | 6 | 24 |
| ECONOMIC | 4 | 23 | 2 | 7 | 2 | | 38 |
| EXTERNAL REGULATION AND REPUTATION | | | | 1 | 13 | | 14 |
| FACTUAL | | 6 | 3 | 1 | 4 | 2 | 16 |
| FAIRNESS & EQUALITY | 20 | 13 | 11 | 4 | | 36 | 84 |
| HEALTH & SAFETY | 8 | 64 | 2 | | | | 74 |
| MORALITY & ETHICS | 12 | 1 | 12 | 2 | 23 | 3 | 53 |
| PERSONAL SYMPATHY & SUPPORT | 2 | 6 | 76 | | 32 | 5 | 121 |
| POLICY DESCRIPTION, PRESCRIPTION,&EVALUATION | 6 | 67 | 20 | 24 | 39 | 3 | 159 |
| POLITICAL FACTORS & IMPLICATIONS | 18 | 47 | 61 | 29 | 24 | 2 | 181 |
| PUBLIC SENTIMENT | 1 | 7 | 27 | 5 | 7 | | 47 |
| QUALITY OF LIFE | 2 | 11 | 11 | 17 | 5 | 1 | 47 |
| SECURITY & DEFENSE | 1 | 2 | 46 | 7 | 56 | | 112 |
| LEGALITY, CONSTITUTIONALITY,&JURISDICTION | 17 | 9 | 11 | 22 | 7 | 9 | 75 |
| **Grand Total** | **98** | **319** | **313** | **151** | **277** | **72** | **1230** |

Fig. 1. Frame v/s Issue distribution for the available labelled data. The colours indicate the unequal distribution as well as the large class imbalance in the data

A similar analysis was conducted over author category of the labelled data. This category information was omitted during training (as the distribution across frames were similar in behaviour.)

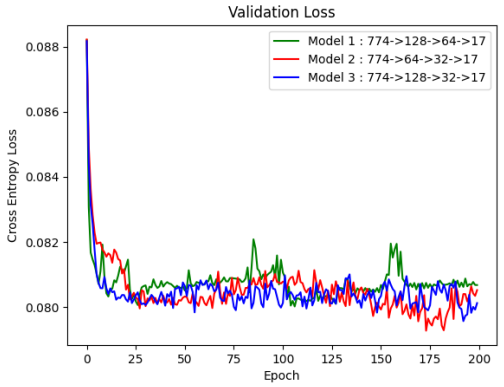
| Count of text | author | | |
|---|---|---|---|
| Frame | democrat | republican | Grand Total |
| (SELF) PROMOTION | 65 | 99 | 164 |
| CAPACITY & RESOURCES | 4 | 9 | 13 |
| CRIME & PUNISHMENT | 3 | 5 | 8 |
| CULTURAL IDENTITY | 20 | 4 | 24 |
| ECONOMIC | 7 | 31 | 38 |
| EXTERNAL REGULATION AND REPUTATION | 2 | 12 | 14 |
| FACTUAL | 14 | 2 | 16 |
| FAIRNESS & EQUALITY | 83 | 1 | 84 |
| HEALTH & SAFETY | 54 | 20 | 74 |
| MORALITY & ETHICS | 25 | 28 | 53 |
| PERSONAL SYMPATHY & SUPPORT | 69 | 52 | 121 |
| POLICY DESCRIPTION, PRESCRIPTION,&EVALUATION | 55 | 104 | 159 |
| POLITICAL FACTORS & IMPLICATIONS | 100 | 81 | 181 |
| PUBLIC SENTIMENT | 36 | 11 | 47 |
| QUALITY OF LIFE | 41 | 6 | 47 |
| SECURITY & DEFENSE | 67 | 45 | 112 |
| LEGALITY, CONSTITUTIONALITY,&JURISDICTION | 36 | 39 | 75 |
| **Grand Total** | **681** | **549** | **1230** |

### B. Model

Input tweet embeddings (768) were obtained from a pre-trained Distilled RoBERTa and the 6 issue categories were one-hot encoded. The baseline model is a feed forward neural network with 2 hidden layers, ReLu activation and Dropout of 0.5 between the hidden layers. Three different architectures were considered and Model 3 was chosen as baseline model for further experiments.

| | Model 1 | Model 2 | Model 3 |
|---|---|---|---|
| F1 Macro | 0.326619 | 0.309662 | 0.335432 |
| F1 Micro | 0.464865 | 0.497297 | 0.481081 |
| F1 Weighted | 0.455197 | 0.465865 | 0.47824 |

Fig. 2. F1 score on validation data using the best model from 3 different model architectures (from multiple experiments). Model 1: hidden(128, 64), Model 2: hidden(64, 32) and Model 3: hidden(128, 32). The models were trained on Cross Entropy loss using Adam optimizer with Train-Validation split of 85:15




## III. Baseline Extensions

A direct method for performance improvement is to fine-tune the pre-trained RoBERTa first on the unlabelled twitter data using the masked objective and then on the labelled data for classification of Frames. However, due to resource constraints and time limitation, this method will not be pursued in this project report. Instead, we focus on improving the model performance by annotating more data for training.

First, we sample from the unlabelled 30000 tweets (annotated with 'issue' category), and demonstrate performance improvements. From this, we infer a relation between the confidence of similarity and the exhibited performance.

Using the threshold from the previous step, we sample data from the unlabelled 3.2 million tweets obtained from https://github.com/alexlitel/congresstweets. We build a model for predicting issue category and annotate the labelled train data and the sampled unlabelled data with prediction over the issue category. This combined data is then used to train a model for Frame classification.

A promising extension to this approach is to perform self distillation after the second satge. Some basic experiments on self distillation using the baseline model was performed, but this did not result in much improvement. This might be because the baseline model overfit on the data and didn't generalise enough for samplign from such a big corpus. Hence, once we have the new model, if we continue to perform self distillation, and combine both the model score as well as the sentence similarity

| Threshold | Trained on labelled + Unlabelled | | | Similarity score given as input | | | | Loss of each instance weighted by similarity | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | F1 Macro | F1 Micro | F1 Weighted | | F1 Macro | F1 Micro | F1 Weighted | | F1 Macro | F1 Micro | F1 Weighted |
| 0.7 | Un a | 0.298967 | 0.491892 | 0.465736357 | UnCS a | 0.314928 | 0.47027 | 0.448008662 | UnCSLoss a | 0.385203 | 0.502703 | 0.498541516 |
| | Un b | 0.346896 | 0.486486 | 0.469289517 | UnCS b | 0.353341 | 0.497297 | 0.492684298 | UnCSLoss b | 0.350036 | 0.502703 | 0.492877852 |
| | Un c | 0.316549 | 0.475676 | 0.458350168 | UnCS c | 0.344529 | 0.491892 | 0.481225221 | UnCSLoss c | 0.323931 | 0.491892 | 0.466081716 |
| 0.65 | Un a | 0.319781 | 0.491892 | 0.472235069 | UnCS a | 0.371333 | 0.486486 | 0.48521251 | UnCSLoss a | 0.350723 | 0.491892 | 0.483912903 |
| | Un b | 0.349582 | 0.508108 | 0.481073438 | UnCS b | 0.315373 | 0.491892 | 0.474164802 | UnCSLoss b | 0.351958 | 0.481081 | 0.470416854 |
| | Un c | 0.323737 | 0.486486 | 0.465852688 | UnCS c | 0.343619 | 0.481081 | 0.471591859 | UnCSLoss c | 0.332931 | 0.491892 | 0.468828374 |
| 0.6 | Un a | 0.307869 | 0.481081 | 0.455731855 | UnCS a | 0.29622 | 0.47027 | 0.450011249 | UnCSLoss a | 0.292395 | 0.459459 | 0.436044795 |
| | Un b | 0.30563 | 0.491892 | 0.468919839 | UnCS b | 0.299633 | 0.459459 | 0.439727045 | UnCSLoss b | 0.305567 | 0.475676 | 0.454822472 |
| | Un c | 0.3201 | 0.486486 | 0.460141285 | UnCS c | 0.312754 | 0.497297 | 0.460415404 | UnCSLoss c | 0.29052 | 0.481081 | 0.457528538 |
| 0.55 | Un a | 0.277043 | 0.497297 | 0.449918406 | UnCS a | 0.292782 | 0.502703 | 0.469115654 | UnCSLoss a | 0.329144 | 0.486486 | 0.469486413 |
| | Un b | 0.29156 | 0.491892 | 0.454924846 | UnCS b | 0.320315 | 0.518919 | 0.49452578 | UnCSLoss b | 0.292149 | 0.491892 | 0.453140111 |
| | Un c | 0.299096 | 0.497297 | 0.46048217 | UnCS c | 0.284173 | 0.513514 | 0.463095634 | UnCSLoss c | 0.310682 | 0.497297 | 0.476902491 |
| 0.5 | Un a | 0.273989 | 0.508108 | 0.455790152 | UnCS a | 0.290327 | 0.508108 | 0.459390599 | UnCSLoss a | 0.306158 | 0.502703 | 0.465969387 |
| | Un b | 0.290156 | 0.524324 | 0.473362672 | UnCS b | 0.304322 | 0.502703 | 0.463167325 | UnCSLoss b | 0.290489 | 0.513514 | 0.466608232 |
| | Un c | 0.300624 | 0.502703 | 0.463485818 | UnCS c | 0.286533 | 0.524324 | 0.468800476 | UnCSLoss c | 0.299153 | 0.513514 | 0.475181847 |

Fig. 3. Sample data from the 30000 unlabelled set with "issue" category after joining on tweet-id with 3.3 million tweets to get the tweet text. Compute cosine similarity with the train data to obtain the similarity scores. Threshold on similarity score and assign label to the unlabelled data using max similarity score. Use 3 different methods as explained in section IV A to train multiple models at different similarity thresholds. The model highlighted in yellow identifies the best model out of the 3 top models for each [method, similarity threshold] combination. From this we see that adding additional training data at a good similarity threshold does improve model performance. But as we decrease the similarity threshold, the performance suffers. Note that for Threshold $< 0.65$, our performance is below the best baseline performance of (0.335, 0.481, 0.478). For later experiments involving full corpus (3.2 million), similarity threshold of 0.7 was used to acquire data.

score to generate additional training data, we should be able to further increase our performance. This would help us create a model for Frame classification that generalizes well. We can also achieve automatic labelling capabilities as manual labelling is quite expensive in terms of human resource (time, effort and money). In this process, we can aggregate more training data for building more complex models that generalize much better and can be deployed in practical applications.

## IV. Experiments

### A. Tweet Similarity Performance Analysis

Data is mined from the unlabelled data using cosine similarity threshold and assigned the label of it's nearest neighbour in the training set. The labelled train data along with the label annotated sampled data were used to train the base model. Refer to Fig 3 description to understand the data mining procedure. The three methods mentioned in it are explained below :

- *Method 1 : Trained on labelled + Unlabelled*
  The model was trained on [text embedding + one-hot issue embedding] using the assigned labels as true label and cross entropy loss.
- *Method 2 : Similarity score given as input*
  The model was trained on [text embedding + one-hot issue embedding + similarity score with nearest neighbour in training set] using the assigned labels as true label and cross entropy loss. Note that for the labelled data, 1 was used as the input similarity score.
- *Method 3 : Loss of each instance weighted by similarity*
  The model was trained on [text embedding + one-hot issue embedding] using the assigned labels as true label and weighted cross entropy loss. The loss of each instance was weighted by it's similarity score and then aggregated over the mini batch to perform gradient update. Note that for the labelled data, 1 was used to weight its loss.

The stats of data mined for each threshold is given in Table I. The learning curves have been omitted from the report due to the large number of plots and page limit constraints.

### B. Data Mining from 3.2 million

The corpus was split into 34 chunks and the tweet embeddings were retrieved for each set. For each tweet in the labelled data, it's nearest neighbours ($> 0.7$ similarity score) were retrieved from the corpus. Retrieving embeddings for each chunk takes about 5 hours on Google cloud CPU machine. For the retrieved tweets, labels were assigned to be the same as their nearest neighbour from the labelled dataset.

Data retrieved from 3.2 million corpus

using similarity threshold of 0.7 = 5546

| Threshold | Data mined |
|---|---|
| 0.70 | 55 |
| 0.65 | 278 |
| 0.60 | 983 |
| 0.55 | 2592 |
| 0.50 | 5311 |

TABLE I
STATS OF DATA MINED FROM THE 30000 UNLABELLED SET USING COSINE SIMILARITY THRESHOLD AND ASSIGNING NEAREST NEIGHBOUR LABEL

| Label | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train | 38 | 13 | 53 | 84 | 75 | 8 | 112 | 74 | 47 | 24 | 47 | 181 | 159 | 14 | 16 | 164 | 121 |
| MinedData | 20 | 84 | 186 | 445 | 299 | 3 | 268 | 344 | 271 | 181 | 425 | 1145 | 234 | | 148 | 177 | 1316 |

Fig. 4. Mined Data stats from 3.2 million tweet data using 0.7 threshold

### C. Issue Category Prediction Model

Using the tweet embedding as input, a model was trained to predict probabilities over the issue categories (abort, aca, guns, immig, isis, lgbt). A feed forward neural network with 3 hidden layers (256, 128, 32) was used along with ReLU activation, Dropout (0.5) betwen each pair of hidden layers, Cross Entropy loss function and Adam optimizer. The training data comprised of 16460 tweets and was split into train, validation and test in the ratio of 80:15:5. The resultant distribution of issue categories for true labels and model predictions on the data used is given in Fig 6 and the model performance is given in Table II.

| Data | F1 macro | F1 macro | F1 macro |
|---|---|---|---|
| Validation | 0.76409 | 0.82746 | 0.82329 |
| Test | 0.75402 | 0.81652 | 0.80973 |

TABLE II
ISSUE CATEGORY PREDICTION MODEL PERFORMANCE : THE HUGE DATA IMBALANCE RESULTS IN THE LOWER F1 MACRO SCORE.

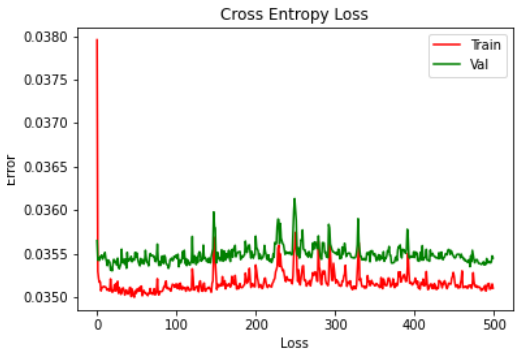



Fig. 5. Learning curve for Issue Category prediction model

| Count of tweet_id | Predicted Labels | | | | | | |
|---|---|---|---|---|---|---|---|
| True labels | abort | aca | guns | immig | isis | lgbt | Grand Total |
| abort | 447 | 19 | | 2 | 54 | 3 | 525 |
| aca | 22 | 2348 | 1 | 29 | 416 | 1 | 2817 |
| guns | 4 | 46 | 1147 | 22 | 411 | 10 | 1640 |
| immig | 1 | 37 | 1 | 866 | 356 | 2 | 1263 |
| isis | 17 | 308 | 134 | 110 | 9082 | 13 | 9664 |
| lgbt | 3 | 5 | 6 | 5 | 137 | 395 | 551 |
| **Grand Total** | **494** | **2763** | **1289** | **1034** | **10456** | **424** | **16460** |

Fig. 6. The ideal distribution would be a diagonal. But we see from the color coding that majority of the predicted labels are correct (green diagonal). Hence this is a good enough model for predicting issue categories for the mined data.

For further experiments, we'll use this model to assign issue category to data for which issue category is not available..

### *D. Frame prediction*

The mined data is passed through the Issue prediction model to obtain issue categories. For the labelled data, the available issue category was one-hot encoded. During training, the weighted cosine similarity Cross Entropy loss function mentioned in Method 3 (section IV A) was used.

The base model was trained using the labelled train data and additionally mined data. This mined data was processed in 3 different methods to improve the model performance.

- Method A : uses the entire 5546 mined data mentioned in section IV B along with the train data to train the model.
- Method B : runs the best model of section IV A (refer Fig 3) on the mined data and get predicted labels. The data for which the predicted label does not match the previous assigned label (from nearest neighbour) is discarded, resulting in 3609 data in addition to train data. The model is trained on this filtered set.
- Method C : To combat class imbalance which has been increased after the addition of the mined data, for each class, the number of extra data added to it is capped. For the purpose of the experiment, a cap size of 100 was used. Refer Fig 7 to see which all class's data got clipped.

| Label | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Train | 38 | 13 | 53 | 84 | 75 | 8 | 112 | 74 | 47 | 24 | 47 | 181 | 159 | 14 | 16 | 164 | 121 |
| Mined : Method A | 20 | 84 | 186 | 445 | 299 | 3 | 268 | 344 | 271 | 181 | 425 | 1145 | 234 | | 148 | 177 | 1316 |
| Mined : Method B | 19 | | 73 | 428 | 75 | | 112 | 275 | 105 | 4 | 194 | 920 | 101 | | | 152 | 1151 |
| Mined : Method C | 20 | 84 | 100 | 100 | 100 | 3 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | | 100 | 100 | 100 |

Fig. 7. Data distribution across labels for Train and mined data

## V. Conclusion

We have demonstrated performance improvement through addition of label annotated data which was acquired through sentence similarity from the corpus of 3.2 million tweets. There is a lot of scope for improving the model via repeated training and re-sampling. Due to time constraints, class weighted loss functions in addition to the similarity weighted loss could not be tested. We'll leave this for the future.

| Model | F1 macro | F1 micro | F1 Weighted |
|---|---|---|---|
| Method A : a | 0.33782 | 0.53513514 | 0.51240496 |
| Method A : b | 0.341808 | 0.53513514 | 0.50461302 |
| Method A : c | 0.366311 | 0.55135135 | 0.52156904 |
| | | | |
| Method B : a | 0.369694 | 0.52972973 | 0.5196309 |
| Method B : b | 0.353628 | 0.51351351 | 0.50454829 |
| Method B : c | 0.359209 | 0.51351351 | 0.49840765 |
| | | | |
| Method C : a | 0.365768 | 0.51891892 | 0.50379862 |
| Method C : b | 0.353608 | 0.51351351 | 0.49896003 |
| Method C : c | 0.387474 | 0.51351351 | 0.50214304 |

Fig. 8. F1 scores on Validation data of top 3 models from trained on the Data : Train + Mined using Methods A, B and C.

## VI. Future work

Given infinite time and computational resources, we can train a RoBERTa or any other sentence encoding model on the entire corpus of twitter data (not limited to the 3.3 million). Once we have a very good twitter text embedding model, we can add additional feed forward layers on top of this and train it for frame classification as we did during the course of the project. The tweet encoder model would give us better estimates of tweet similarity than using a pre-trained sentence embedding. Hence we should be able to observe much higher performance improvements after using the mined data. We can further follow up using self distillation techniques and keep improving the classifier.

## VII. Appendix

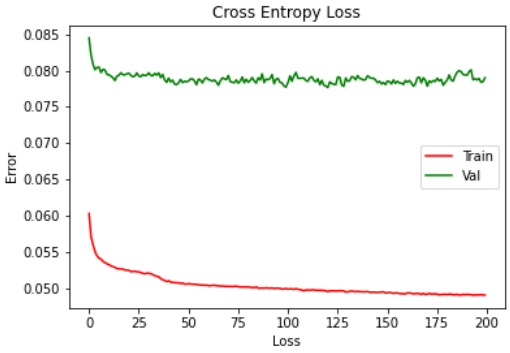



Fig. 9. Learning curve of Method A model c

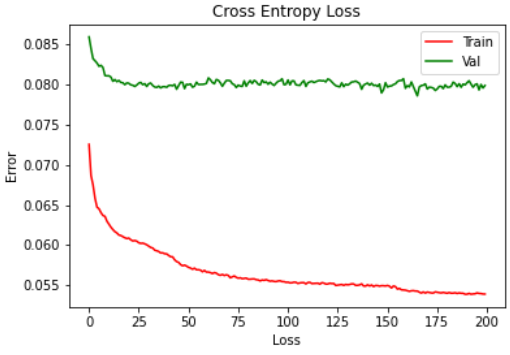



Fig. 10. Learning curve of Method B model c