CS779 Competition: Sentiment Analysis

Edula Vinay Kumar Reddy
241110024
vinaykr24@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

The task given in the sentiment analysis competition is to classify text into three classes: positive, negative, and neutral. I tried several models, including RNNs, Transformer Encoder, and hybrid architectures like RCAN. For dealing with imbalanced data, I have used Focal Loss. I have created custom embeddings combining linguistic features such as POS tags, dependency tags, negation, and GloVe embeddings. My final solution was to ensemble three models with majority voting: RNN with Focal Loss, RCAN, and BiLSTM with Global Max Pooling. With this approach, my model achieved 1st rank in the competition with a F1-score of 69.7%.

1 Competition Result

Codalab Username: E_241110024
Final leaderboard rank on the test set: 1
F1 Score wrt to the best rank: 69.7%

2 Problem Description

This is a Sentiment Analysis problem, which is a three-class classification task: Positive, Neutral, and Negative. This is an important problem in various sectors to understand user behavior, as it gives insight into opinions and emotions expressed in text. To solve this problem we should come up with a model which is capable of processing different kinds of text content. The challenge is, however, to handle language related problem such as sarcasm, negation, and different text structures.

3 Data Analysis

1. **Training Data Description:** The dataset has 92,228 entries in the `train.csv` dataset. There are three columns: `text_id`, `sentence`, and `gold_label`.
   - The `text_id` column contains a unique ID for every row. The `sentence` column contains text data, most of which are conversational exchanges between various people.
   - The `gold_label` column is the target variable for sentiment classification, with values: 1: Positive sentiment, 0: Neutral sentiment, -1: Negative sentiment

2. **Analysis and Insights About the Data:**
   1. **Missing Values:** It is a complete dataset, and no column or row contains missing values.
   2. **Class Distribution:**The class distribution shows an imbalance.This imbalance may affect model performance and might require techniques to address it.
      - Class 0: **45,489 samples** (majority)
      - Class 1: **27,353 samples**
      - Class -1: **19,386 samples** (minority)

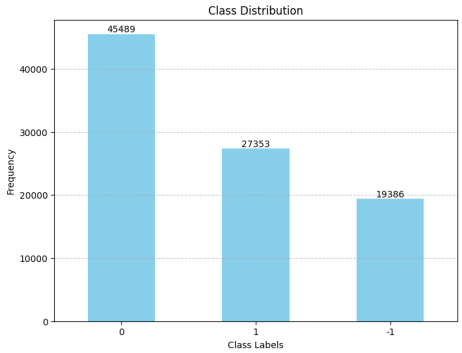

Figure 1: Classes Distribution

3. **Class-wise Sentence Length Distribution:** The sentence length distributions for all classes (-1, 0, and 1) are similar, with the majority clustered around shorter lengths. Longer sentences exist but are negligible. There appears to be no relationship between sentence length and sentiment for this task, based on the analysis.

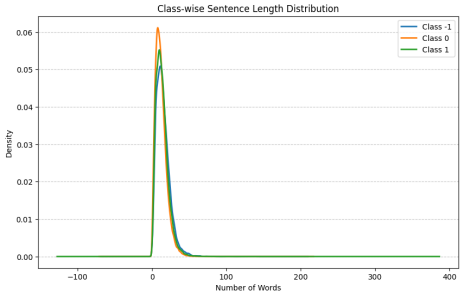

Figure 2: Class-wise Sentence Length Distribution

4. **Bigrams:** During tokenization, using some bigrams as single token can capture meaningful combinations like “not good” or “very bad,” which are crucial for sentiment analysis.

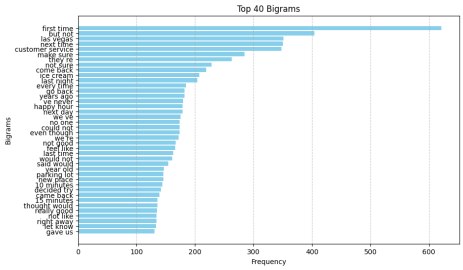

Figure 3: Top 40 Bigrams in the Data

5. **Corpus-wide Sentence Length Distribution:** The corpus-wide sentence length distribution helps determine the maximum sentence length during training and inference.

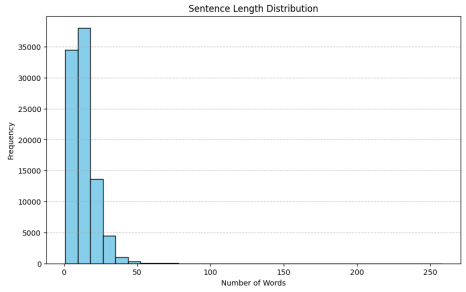

Figure 4: Corpus-wide Sentence Length Distribution

- **Test Data:**
  The test dataset contains **5,110 rows**, which is approximately **5.54%** of the train dataset size (92,228 rows). It consists of the columns:
  - `text_id`: Unique identifier for each row
  - `sentence`: Text data for prediction

4 Model Description

Model-1: Transformer Encoder with Attention

- **Preprocessing:**
  - Sentences are lemmatized using spaCy. Words with a frequency of 1 are removed to clean the vocabulary[3]. Extracted bigrams and trigrams with a minimum frequency of 5 and created a combined vocabulary of unigrams, bigrams, and trigrams. Next sentences are tokenized into sequences by hierarchically matching trigrams, bigrams, and unigrams.
- **Model Architecture:**
  - Embedding layer initialized with pre-trained GloVe vectors.
  - 4 Transformer layers with 2 attention heads, Feedforward layer (hidden size: 256), Layer normalization and residual connections.
  - Fully connected layer for classification into 3 sentiment classes.
- **Training Details:**
  - Loss function: CrossEntropy.
  - Optimizer: Adam with weight decay.
  - Learning rate scheduler: ReduceLROnPlateau.
- **Performance:**
  - F1-score on Dev Set: 64.6%.

Model-2: BiLSTM with Attention

- **Preprocessing:**
  - Sentences are lemmatized and stopwords were removed using spaCy.Custom positivity scores are calculated for words. Scores for words are derived based on word associations with sentiment classes.
- **Model Architecture:**
  - **Embedding Layer:** 100-dimensional GloVe embeddings for words + 50-dimensional POS embeddings + 50-dimensional embeddings from scaled positivity scores.
  - **BiLSTM:** Uses Combined 200-dimensional embeddings. Have Hidden dimension of 128. Output is passed to an attention mechanism.
  - **Attention Layer:** Uses self-attention to compute a context vector for classification.

  - **Output Layer:** Fully connected layer maps to 3 sentiment classes.
- **Training Details:**
  - Loss function: CrossEntropy.
  - Optimizer: Adam with weight decay.
  - Learning rate scheduler: ReduceLROnPlateau.
  - Early stopping is applied with patience of 5 epochs.
- **Performance:**
  - F1-score on Dev Set: 65. 6%.

Model-3: BiLSTM with Global Max Pooling

- **Preprocessing:** Described in section 5.1
- **Model Architecture:**
  - **Embedding Layer:** Input dimension: 264 (combined features).
  - **BiLSTM:** Hidden size of 256 with 2 bidirectional layers with dropout (0.5).
  - **Global Max Pooling:** Captures the most important features across time steps.
  - **Output Layer:** Fully connected layer maps to 3 sentiment classes.
- **Training Details:**
  - Loss function: CrossEntropy.
  - Optimizer: Adam with weight decay.
  - Learning rate scheduler: ReduceLROnPlateau.
  - Early stopping applied with patience of 5 epochs.
  - Gradient clipping used with a maximum value of 5.
- **Performance:**
  - F1 score on test Set: 69%.

Model-4: RCNN with Focal Loss

- **Preprocessing:** Described in section 5.1
- **Model Architecture:**
  - **Embedding Layer:** Input dimension: 264 (combined features).
  - **Bidirectional LSTM:** Hidden size of 512. Outputs bidirectional context features for each token.
  - **Convolutional Layer:** Applies 1D convolution with kernel size 3 over LSTM outputs and extracts high-level features from the context.
  - **Max Pooling:** Captures the most prominent feature for each filter across the sequence.
  - **Output Layer:** Fully connected layer maps pooled features to 3 sentiment classes.
- **Focal Loss:**
  - A specialized loss function is adapted to handle class imbalance by emphasizing hard-to-classify examples[4] .
  - **Parameters:**
    - **Alpha (α):** Balancing factor to adjust the importance of classes. Here, $\alpha = 1$.
    - **Gamma (γ):** Focusing parameter to reduce the weight of well-classified examples. Here, $\gamma = 2$.
  - **Mathematical Representation:**

$$FL(p_t) = -\alpha \cdot (1 - p_t)^{\gamma} \cdot \log(p_t)$$

    where p_t is the predicted probability for the true class.Focal Loss improves the model's performance by prioritizing hard-to-classify samples, reducing the impact of class imbalance.

- **Training Details:**
  - Loss function: Focal Loss (as described above).
  - Optimizer: Adam with weight decay.
  - Learning rate scheduler: ReduceLROnPlateau.
  - Early stopping applied with patience of 5 epochs.
  - Gradient clipping used with a maximum value of 5.
- **Performance:**
  - F1 score on test Set: 67.9%.

Model-5: Recurrent Convolutional Attention Network (RCAN)[1]

- **Preprocessing:** Described in section 5.1
- **Model Architecture:**
  - **Embedding Layer:** Input dimension: 264 (combined features).
  - **Bidirectional LSTM:** Hidden size: 256, Outputs bidirectional context features for each token.
  - **Convolutional Layer:** Applies 1D convolution with kernel size 3 over LSTM outputs and extracts localized features from the LSTM output.
  - **Attention Mechanism:** Computes attention scores to emphasize critical parts of the sequence and generates a weighted feature vector based on attention scores.
  - **Output Layer:** Fully connected layer maps attention-enhanced features to 3 sentiment classes.
  - **Regularization:** Dropout layer applied to prevent overfitting.
- **Training Details:**
  - Loss function: CrossEntropyLoss for multi-class classification.
  - Optimizer: Adam with weight decay.
  - Learning rate scheduler: ReduceLROnPlateau.
  - Early stopping applied with patience of 5 epochs.
  - Gradient clipping used with a maximum value of 5.
- **Performance:**
  - F1-score on test set: 69%.

Final Model: Majority Voting Ensemble

- Combined predictions from three models:
  - * **Recurrent Convolutional Attention Network (RCAN):** Captures sequential, local, and sentiment-rich features using LSTM, convolution, and attention mechanisms.
    * **RCNN with Focal Loss:** Handles class imbalance by focusing on hard-to-classify examples using Focal Loss.
    * **BiLSTM with Global Max Pooling:** Extracts sequential features and summarizes key information using pooling.
  - Implemented a majority voting strategy to make final predictions based on the outputs of these three models.
- **Majority Voting Strategy:** For each test example, predictions from the three models were aggregated. The final label was assigned as the label predicted by the majority of the models.
- **Advantages:**: Combines the strengths of individual models sequential modeling, localized feature extraction, and dynamic attention mechanisms.
- **Performance:** F1-Score 69.7% on Test dataset which is greater than individual models score. This concludes that, even if one model is making any errors, other models correct those errors.

5 Experiments

1. Data Pre-processing

- Tokenized sentences using SpaCy with a maximum length of 64 tokens.
- Used pre-trained GloVe embeddings (100-dimensional) for word representation and assigned random embeddings to unseen words..
- Added linguistic features to embeddings:
  - **POS Embeddings:** Learnable 30-dimensional embedding for each type of POS tag (NOUN, ADJ..), to capture syntactic roles.
  - **Dependency Embeddings:** Learnable 30-dimensional embedding, encoding grammatical relations. This tag indicate role played by the word like subject, object etc.
  - **Positivity Scores:** Developed a custom positivity score for each word based on its frequency in the positive class within the dataset, reflecting the degree of positivity the word conveys.
  - **Negation:** Compiled a list of words that negate meaning and checked if the current word had any of these negation words as an ancestor. If found, a binary feature was used to indicate this.
  - **Punctuation, and Capitalization Features:** Binary features indicating punctuation and capitalization.
  - **Contextual Embeddings:** Averaged GloVe embeddings of dependent tokens for enhanced context.

- **Reason for Choosen Pre-processing:** My idea for this preprocessing is to enrich the data and make it more informative for the model to find the sentiment. Pre-trained GloVe embeddings provide a good starting point for the model in understanding word meanings. Adding features like POS and dependency embeddings help the model understand the role of each word in a sentence, such as whether it's a noun or subject. Positivity scores provide information on the likelihood of a word to carry positive sentiment. Features for negation, punctuation, and capitalization help the model pick up on shifts in meaning, emphasis, or tone. Averaging the embeddings of related words adds context to each word, helping to improve the model's understanding of how words work together.

2. Training Procedure[Final model]

- **Optimizer:** Adam optimizer with weight decay 10^{-5}.
- **Learning Rates:** Initial learning rate of 0.001, reduced dynamically using ReduceLROnPlateau scheduler with patience of 5 epochs.
- **Epochs:** All models trained for up to 40 epochs, with early stopping triggered after 5 epochs of no improvement in validation F1-score.
- **Batch Size:** 64 for all models.
- **Training Time:** 10 minutes for loading glove embedding and data preprocessing, 2 minutes per epoch in GPU for BiLSTM and RCNN with Focal Loss and 4 minutes per epoch for RCAN.
- **Loss Functions:**: CrossEntropyLoss for BiLSTM and RCAN and Focal Loss ($\alpha = 1$, $\gamma = 2$) for RCNN to handle class imbalance.
- **Gradient Clipping:** Applied with a max norm of 5 to stabilize training.

3. Hyper-parameters for Different Models

Selection Process: Hyper-parameters were selected by manually setting and experimenting various combination of values and verifying validation F1-score. Hidden sizes, dropout rates, and loss functions were adjusted by keeping in mind both performance and computational efficiency.

Model	Hyper-parameter	Value	Reason for Selection
BiLSTM	Hidden Size	128	Lightweight, fast convergence.
	Dropout	0.5	Prevents overfitting.
	Pooling	Global Max	Summarizes critical features.
RCNN	Hidden Size	512	Captures complex patterns.
	Dropout	0.5	Reduces overfitting.
	Kernel Size	3	Captures local patterns.
	Loss Function	Focal Loss	Addresses class imbalance.
RCAN	Hidden Size	256	Balanced complexity.
	Dropout	0.5	Prevents overfitting.
	Kernel Size	3	Extracts n-gram features.
	Attention	Yes	Focuses on sentiment-rich tokens.

Table 1: Hyper-parameters for different models

6 Results

1. **Results of Different Models on Dev Data:**

Model	F1-Score	Rank
Transformer Encoder with Attention	64.6%	6
BiLSTM with Attention	65.6%	5
BiLSTM + Global Max Pooling	67%	2
RCNN + Focal Loss	66.9%	3
Majority Voting Ensemble(Final model)	68.1%	1

Table 2: Performance of Created Models on Dev Data

2. **Results of Different Models on Test Data:**

Model	F1-Score	Rank
BiLSTM + Global Max Pooling	69%	1
RCNN + Focal Loss	67.9%	3
RCAN	69%	1
Majority Voting Ensemble(Final model)	69.7%	1

Table 3: Performance of Created Models on Test Data

3. **Explanation of Results:** Majority Voting Ensemble achieved the best results on both the dev and test datasets, with the highest F1-score of 68.1% on dev (RCNN + RCNN Focal loss + BiLSTM with Global max pooling) and 69.7% on test (RCNN with Focal loss + RCAN +.BiLSTM with Global max pooling). The RCAN model combined the Bidirectional LSTM with localized feature extraction using convolutional layers, and attention mechanisms that dynamically focus on parts of the vector which tells about sentiment.The ensemble approach further boosted performance by leveraging the strengths of all three models.

7 Error Analysis

1. **Error Analysis on Dev and Test Sets:** Sentences which belongs to neutral class and slightly looks like positive or negative sentiment are misclassified by the model most of the time.
2. **Class Imbalance:** Need to use techniques like data augumentation because despite using Focal Loss, under-represented classes like positive sentiments were sometimes misclassified.

3. **Suggestions for Improvement:**
    - Incorporating external knowledge bases or pre-trained contextualized embeddings (e.g., BERT) could improve understanding of ambiguous or nuanced text.
    - Data augmentation techniques to balance the dataset further.
4. **Interesting Insights:** Due to use of custom word positivity score, sentences with explicit sentiment-bearing words (e.g., "amazing," "terrible") were classified accurately across all models and models are making misclassifications when statements are subjective, such as - I am okay, which could be interpreted as positive or neutral depending on context.

8 Conclusion

In this competition i have experimented with various models for sentiment classification, Finally i used Majority Voting Ensemble that achieved the best results on both the dev and test datasets. Combining linguistic features, contextual information, and diverse model architectures helped in achieving best classification performance. Recurrent Convolutional Attention Network (RCAN) emerged as the best-performing individual model (F1-Score 69% on Test set) , by combinedly using the advantages of sequential modeling, convolutional feature extraction, and attention mechanisms to handle complex sentences and sentiment-rich text.The Majority Voting Ensemble combined the strengths of all models, improving robustness and achieving the highest F1-score. Despite these efforts, model is struggling with ambiguous sentences and classifying neutral classe. Further work should be to tackle this problem.

Recommendations

- For similar sentiment classification tasks, an ensemble approach like Majority Voting can improve generalization by utilizing advantages of multiple architectures.
- Using Focal Loss or other imbalance-handling techniques is essential when dealing with datasets with under-represented classes.

Future Directions

- Experimenting with data augmentation techniques to address class imbalance problem.
- To Work with pre-trained transformer-based models like BERT or RoBERTa to enhance contextual understanding.
- Try to predict not only sentiment but other linguistic attributes like emotion or intent which helps in predicting accurate.

References

[1] M. Usama, B. Ahmad, A. P. Singh, and P. Ahmad, "Recurrent Convolutional Attention Neural Model for Sentiment Classification of Short Text," in *2019 International Conference on Cutting-edge Technologies in Engineering (ICon-CuTE)*, 2019, pp. 40–45. [Online]. Available: `https://doi.org/10.1109/ICon-CuTE47290.2019.8991445`.

[2] S. M. Rezaeinia, R. Rahmani, A. Ghodsi, and H. Veisi, "Sentiment analysis based on improved pre-trained word embeddings," *Expert Systems with Applications*, vol. 117, pp. 139–147, 2019. [Online]. Available: `https://doi.org/10.1016/j.eswa.2018.08.044`.

[3] H. Saif, M. Fernandez, Y. He, and H. Alani, "On Stopwords, Filtering and Data Sparsity for Sentiment Analysis of Twitter," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 810–817. [Online]. Available: `http://www.lrec-conf.org/proceedings/lrec2014/pdf/292_Paper.pdf`.

[4] P. Basu, S. Tiwari, J. Mohanty, and S. Karmakar, "Multimodal Sentiment Analysis of #MeToo Tweets using Focal Loss (Grand Challenge)," in *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*, 2020, pp. 461–465. [Online]. Available: https://doi.org/10.1109/BigMM50055.2020.00076.