Cyberbullying Detection Identifying using Machine Learning

LIN Jiatong 2230026094
TAO Manxi 2230026145
XIA Yixin 2230026171

***Abstract*—This project focuses on cyberbullying detection. Considering the severe harm of cyberbullying to victims and the deficiencies of existing detection methods, the aim is to improve the accuracy of the detection model. By collecting a dataset with a specific structure, comprehensive data preprocessing is carried out, including cleaning, feature extraction and selection, standardization, and splitting. Multiple methods are used to build and improve the model, such as Naïve Bayes weighted averaging, adversarial testing, and model switching (N - BEATS), and the introduction of Transformer is planned. Experiments are conducted to compare different models and techniques and evaluate their performance differences. The results show that the improvement measures effectively enhance the model's ability to classify sentences containing sensitive words and reduce the misclassification rate. This research provides a more reliable method for cyberbullying detection and contributes to creating a healthy online environment.**

***Index Terms*—Cyberbullying Detection, Machine Learning, NLP, N-Beats, Transformer, Word2Vec**

I. Introduction

A. Research Background

Cyberbullying has become a serious social problem. It is implemented through electronic information communication channels and includes forms like verbal insults, relationship exclusions, and harassments. Victims suffer from psychological traumas such as anxiety and depression, and may even have suicidal tendencies. It also leads to discrimination and violent conflicts at the social level. At the same time, hate speech on the Internet targets specific groups, aggravating social division and instability. With the wide spread of the Internet, cyberbullying is becoming more and more common, and effective detection and prevention are urgently needed.

B. Research Significance

1) Social Level: It helps to create a healthy, harmonious, and inclusive online environment, reduces social contradictions and conflicts caused by cyberbullying and hate speech, promotes mutual understanding and respect among different groups, and maintains social stability and fairness.

2) Individual Level: It can timely detect and intervene in cyberbullying behaviors, provide support and protection for victims, help them avoid or reduce psychological harm, and restore normal study, work, and social life.

C. Research Status

Currently, there are various methods for cyberbullying detection. Some studies combine text features with machine learning algorithms and use user psychological features to improve detection effects or adopt ensemble learning to enhance model performance. However, challenges still remain. For example, models are overly sensitive to specific sensitive words, lack accurate understanding of complex semantics and contexts, and may misclassify sentences containing sensitive words. Moreover, the performance of existing methods varies on different datasets.

D. Research Objectives

- Improve the accuracy of cyberbullying detection models, especially for identifying cyberbullying remarks containing sensitive words and with complex semantics.
- Compare the performance of traditional supervised models and ensemble learning models in cyberbullying detection to determine the best model combination or method.
- Deeply understand the role and limitations of different natural language processing techniques in cyberbullying detection and provide references for subsequent research.

E. Research Methods

1) Data Processing:

- Data Collection: Use a comprehensive dataset containing aggressive and non-aggressive remarks, such as "Aggressive_All.csv" and "Non_Aggressive_All.csv", to ensure data balance.
- Preprocessing: Perform data cleaning (removing numbers, filtering stopwords, removing punctuation, stemming, and word segmentation), feature extraction (TF-IDF transformation), feature selection and data standardization (normalization and standardization) based on information gain, and split the data into training and test sets in an 8:2 ratio.

2) Model Building and Improvement:

- Adopt traditional supervised models (such as the Naïve Bayes weighted averaging method) and ensemble learning models, and use deep feedforward neural networks for training and prediction.
- Improve the model by addressing its limitations, including adversarial testing (balancing samples with sensitive words), switching to the N-BEATS model for dynamic

word embedding generation, and introducing the Transformer model (with multi-head attention mechanism) to enhance semantic understanding ability.

II. Data Sources and Processing

The characteristics of cyberbullying are aggressive, repetitive, and intentional communication among peers. However, most existing cyberbullying detection datasets only focus on aggressive text and classify it as either aggressive or non aggressive, while ignoring the other three aspects of cyberbullying: repetition, peer behavior, and intent to harm. This dataset we used is a comprehensive dataset generated from these four features, to avoids the text having only positive and negative attributes The dataset comprises two subsets: aggressive and non-aggressive text, each containing 118,829 samples, balanced in size with columns labeled *No.* and *Message*.

A. Characteristics of the Dataset

Aggressive Data:

- 40% contains offensive language.
- 30% represents biased opinions on sensitive topics (e.g., race, religion, politics).
- The remaining samples include nonsensical or ambiguous phrases.

Non-Aggressive Data:

- Includes neutral or non-offensive comments.
- Contains ambiguous words or phrases with unclear intent.

B. Dataset Partitioning

The dataset was split into:

- 70% Training Set for model development.
- 15% Testing Set for performance evaluation.
- 15% Validation Set for hyperparameter tuning.

C. Data Preprocessing

We applied the following preprocessing steps to reduce noise and extract meaningful features:

- **Digit Removal:** Eliminated numerical data to avoid irrelevant information.
- **Stop Word Filtering:** Removed high-frequency, low-information words like "is," "the," and "a."
- **Punctuation Removal:** Stripped punctuation marks to ensure clean tokenization.
- **Stemming:** Reduced words to their root forms for enhanced semantic generalization (e.g., "running" $\rightarrow$ "run").
- **Word Segmentation:** Tokenized sentences into individual words for feature extraction.
- **Unified capitalization:**Eliminate text inconsistency caused by case differences, making data more standardized and consistent.
- **Word form reduction:**Resolve the issue of different word deformations in text and improve the accuracy of text mining and information retrieval.

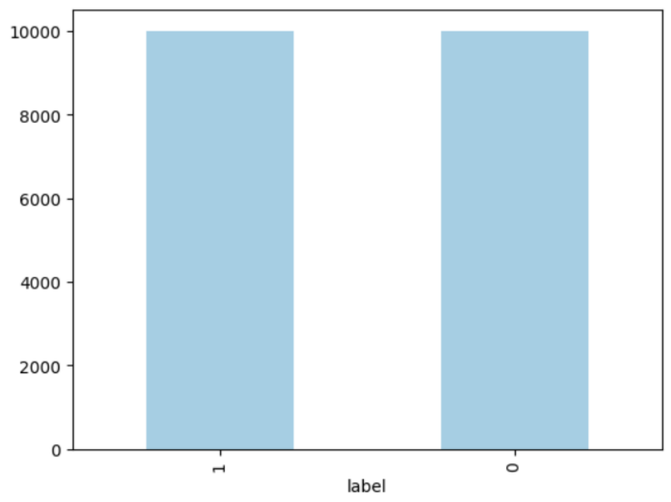

Fig. 1. Aggressive and Non-aggressive data are same size. The data is balanced.

- **Remove empty sentences:**Reduce invalid information in data, lower data noise, and improve data density and quality.
- **Remove duplicate lines:**Avoid the adverse effects of duplicate data on the analysis results.

D. Training Set Bayes Analyses

Fig 1. by examining the shape of the training set, we can understand the data distribution and determine whether there are imbalance issues that might affect the model's learning process. Data imbalance may lead to the model being overly biased towards categories with a larger sample size, thereby reducing the accuracy of predictions for minority categories. It will make it difficult for the model to fully capture the intrinsic features and patterns of various categories of data during the learning process, thereby affecting the model's generalization ability.

III. Methodology

In our research, we implemented multiple models and techniques to address the task of cyberbullying detection, including pre-trained transformer models, feedforward neural networks, and innovative training strategies. This section details the key components of our methodology.

A. TF-IDF + Machine Learning Models

We used **TF-IDF (Term Frequency-Inverse Document Frequency)** to extract feature matrices:

$$TF - IDF(t, d) = TF(t, d) \cdot IDF(t) \tag{1}$$

where:

$$IDF(t) = \log \frac{N}{n_t} \tag{2}$$

Here, $TF(t, d)$ represents the frequency of term t in document d, N is the total number of documents, and n_t is the number of documents containing t.

We applied **PCA (Principal Component Analysis)** for dimensionality reduction and normalized the data before training models such as:

- Support Vector Machine (SVM)
- Random Forest (RF)
- Logistic Regression (LR)

B. Word2Vec + NBLCR + Multilayer Perceptron (MLP)

To capture semantic relationships, we replaced TF-IDF with **Word2Vec embeddings**, which predict a word's context based on its neighbors:

$$P(w_{t+k}|w_t) = \frac{\exp(v_{w_{t+k}}^\top v_{w_t})}{\sum_{w=1}^{W} \exp(v_w^\top v_{w_t})} \tag{3}$$

Here, w_t is the target word, w_{t+k} is a context word, and v_w represents the word vector.

To address equal weighting limitations in Word2Vec, we introduced **NBLCR (Naive Bayes Log Count Ratio)** weights embeddings are computed to adjust word importance:

- Word Frequency Calculation: Counts occurrences of words in each class (aggressive and non-aggressive).
- Log Ratio Computation:

$$w_w = \log \frac{P(w|Class1)}{P(w|Class2)} \tag{4}$$

where:

$$P(w|Class) = \frac{f_c(w) + 1}{\sum_{w'} f_c(w') + |V|} \tag{5}$$

- Applies Laplace smoothing to avoid zero probabilities.

Combines Word2Vec embeddings with NBLCR weights to compute weighted average sentence vectors. The weighted average vector for a sentence is computed as:

$$\mathbf{s} = \frac{\sum_{i=1}^{N} \left(\log \frac{P(w_i|c_1)}{P(w_i|c_2)} \right) \cdot \mathbf{v}_i}{\sum_{i=1}^{N} \left(\log \frac{P(w_i|c_1)}{P(w_i|c_2)} \right)} \tag{6}$$

Where: $\mathbf{v}_i$ is the Word2Vec embedding vector for the i-th word.

Then e used a **Multi-Layer Perceptron (MLP)** for classification:

- **Input Layer**: Takes pre-computed embeddings (e.g., Word2Vec or N-BEATS output) of size 100.
- **Hidden Layers**:
  - Layer 1: Fully connected, 256 neurons, ReLU activation.
  - Layer 2: Fully connected, 128 neurons, ReLU activation.
  - Dropout layers (50%) after each hidden layer to prevent overfitting.
- **Output Layer**: Fully connected layer with 2 neurons for binary classification using softmax activation.

C. Adversarial Training and N-BEATS

So far, our model has achieved high accuracy. However, while maintaining high accuracy, we found that it is very sensitive to some extreme words, such as "fuck" or "stupid," which have strong negative connotations. It still detects them as aggressive text, So we are trying to make some adjustments to the model.

Adversarial Training is a technique originally proposed to improve the robustness of machine learning models by exposing them to adversarial examples. Adversarial examples are carefully crafted inputs that are designed to exploit weaknesses in the model, leading to misclassification. By incorporating such examples into the training data, the model can learn to focus on the semantic meaning of the input rather than being overly influenced by superficial features like specific keywords or noise.

In our case, the adversarial examples are sentences where extreme or offensive words (e.g., "fuck," "stupid") appear in positive or neutral contexts. For instance:

- *"You're so fucking beautiful"* contains an offensive word, but the overall sentiment is positive.
- *"This shit is amazing"* also includes a strong word but has a positive meaning.

Traditional models often misclassify such sentences as aggressive because they overly rely on the presence of offensive words without understanding the broader context.

N-BEATS (Neural Basis Expansion Analysis for Time Series) is a time-series forecasting model designed to decompose input sequences into trend and seasonal components. Although originally developed for time-series analysis, N-BEATS can be adapted to dynamically generate context-aware embeddings for text. The model operates with a stack of fully connected layers, where each block iteratively refines the input by learning trends and relationships.

Why Use These Techniques?: The main motivation behind using adversarial training and N-BEATS is to address the following challenges:

- Sensitivity to Keywords: Traditional models overfit to the presence of offensive words, misclassifying sentences based on isolated terms rather than semantic meaning.
- **Static Word Embeddings**: Pre-trained word embeddings like Word2Vec do not adapt dynamically to sentence context, limiting their ability to capture nuanced relationships.
- **Semantic Robustness**: N-BEATS introduces a dynamic mechanism to adjust word embeddings based on the overall sentence structure and meaning, allowing the model to distinguish between offensive and non-offensive uses of the same word.

How We Implemented It: **Adversarial Training:**

- We manually generated a set of adversarial samples where sensitive words appeared in positive or neutral contexts.
- These samples were added to the training dataset, ensuring that the model was exposed to a variety of sentence structures and contexts.

- During training, we monitored the model's performance on a validation set containing similar adversarial examples to fine-tune hyperparameters.

N-BEATS for Dynamic Embeddings:

- Sentences were first tokenized and converted into numerical sequences using pre-trained Word2Vec embeddings.
- These embeddings were fed into the N-BEATS model, which processed them as sequential data to learn semantic trends and relationships.Key hyperparameters:
  - **Input Length**: Tokenized sequences of 64 tokens.
  - **Learning Rate**: 1×10^{-3}.
  - **Hidden Layers**: Configured N-BEATS blocks to adjust embeddings dynamically based on sentence context.
  - **Loss Function**: Mean squared error for embedding adjustment.
- The backcast output of N-BEATS, representing a refined version of the input sequence, was used as the final sentence representation.
- These dynamic embeddings were then passed to a fully connected neural network for classification.

D. Transformer with Multi-Head Attention

The **Transformer model** has attention mechanism, which enables the model to focus on the most relevant parts of the input. The core of the Transformer is the **Multi-Head Attention** mechanism, which computes a weighted representation of the input based on the relationships between all tokens in a sentence.

For a given input sequence, Multi-Head Attention computes the attention score as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{7}$$

where:

- Q, K, V are query, key, and value matrices derived from the input embeddings.
- d_k is the dimensionality of the key.
- The softmax function ensures that the attention scores sum to 1, highlighting the most relevant parts of the input.

Multi-Head Attention extends this mechanism by splitting the embedding space into multiple subspaces, allowing the model to attend to different aspects of the input simultaneously:

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W_O \tag{8}$$

where each head $head_i$ is computed as:

$$head_i = Attention(QW_{Q_i}, KW_{K_i}, VW_{V_i}) \tag{9}$$

and $W_{Q_i}, W_{K_i}, W_{V_i}, W_O$ are learnable projection matrices.

Why Use Transformers?: Transformers address several limitations of traditional models:

- **Parallel Processing**: Unlike RNNs, Transformers process the entire input sequence in parallel, making them computationally efficient for large datasets.
- **Long-Range Dependencies**: The attention mechanism enables Transformers to capture both local and global relationships within the text, which is essential for understanding complex sentences.
- **Semantic Focus**: Multi-Head Attention allows the model to attend to multiple semantic aspects of the input simultaneously, improving its ability to distinguish between subtle contextual differences.

How We Implemented It:

- **Base Architecture**:
  - 12 Transformer layers (encoder-only architecture).
  - Multi-Head Attention mechanism with 12 attention heads.
  - Hidden size of 768, with a total of 110 million parameters.
- **Classification Head**: A fully connected linear layer appended to the final hidden state of the [CLS] token, producing logits for binary classification.

Pre-trained Weights: We leveraged BERT's pre-trained weights, trained on massive corpora (e.g., BookCorpus and Wikipedia), to initialize the model. This allowed for transfer learning, enabling the model to generalize language understanding to our specific task.

Input Format: The text was tokenized using the BERT tokenizer (`bert-base-uncased`) with:

- **Tokenization**: Splits sentences into subword tokens.
- **Padding and Truncation**: All inputs were padded or truncated to a fixed length of 64 tokens.
- **Special Tokens**: Added [CLS] and [SEP] tokens to mark the start and end of the sentence.
- **Input Representation**: Includes token IDs, attention masks, and optional segment IDs.

E. Optimizer and Learning Rate

We employed the **AdamW optimizer**. Adamw is an improved Adam optimizer, with the main difference being that w represents ”weight decay”, which can effectively avoid model overfitting. Parameter settings for optimizer:

- **Learning Rate**: 3×10^{-5}.
- **Weight Decay**: 0.01.

AdamW combines momentum and adaptive learning rate (Adam) with weight decay (W)and combines adaptive learning rates with L2 regularization, commonly used in NLP and deep learning tasks. AdamW is particularly effective in handling complex models such as BERT, and can better optimize the parameters in the network.

F. Loss Function

The **CrossEntropyLoss** function was used as the objective for classification:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{10}$$

where:

- y_i is the true label.

- $\hat{y}_i$ is the predicted probability for the correct class.

This function measures the divergence between predicted and true label distributions, making it suitable for binary and multi-class classification. The cross entropy loss function calculates the loss value based on the model's outputs (logits) and true labels (1abels). The output of the BERT model is unactivated logits, and the cross entropy loss will automatically apply the softmax activation function to these logits.

G. Training and Evaluation Loops

Training Loop: The training loop involved:

- Forward pass: Input text was tokenized and passed through the BERT model to compute logits.
- Backward pass: Gradients were calculated, and the optimizer updated model parameters.
- Metrics: Tracked training loss and accuracy for each epoch.

Evaluation Loop:

- Computed validation loss and accuracy at the end of each epoch.
- Used `torch.no_grad()` to prevent gradient computation during evaluation, reducing memory usage.

H. Early Stopping Mechanism

To prevent overfitting, an early stopping mechanism was employed:

- **Patience**: Training was halted if validation accuracy did not improve for 3 consecutive epochs.
- **Impact**: Preserved the best-performing model for inference, ensuring efficiency and generalization.

I. K-Fold Cross-Validation

We used **k-fold cross-validation** with $k = 5$ to evaluate model robustness:

- Split the dataset into 5 folds.
- Used 4 folds for training and 1 fold for validation.
- Reported average accuracy, precision, recall, and F1-score across all folds.

IV. Experimental Results

We evaluated our methods using accuracy, precision, recall, f1-score and adversarial Text analysis.

A. Performance Comparison Across Models

The accuracy metric shows the overall performance of each model in classifying offensive and non-offensive text:

- **SVM, LR, RF**: These traditional machine learning models achieve accuracy around 80%-89%, with RF performing better than SVM and LR due to its ability to capture non-linear relationships.
- **Word2Vec + NBLCN + MLP**: Achieves **85.6% accuracy** on average across k-folds, indicating an improvement over traditional models by incorporating word embeddings and deep learning.

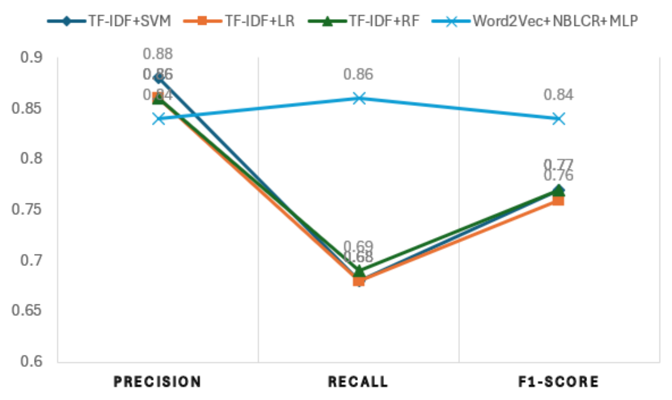

Fig. 2. Compare models with precision, recall and F1-Score

- **Beat Model**: With an accuracy of **97%**, this model significantly outperforms the others, demonstrating its capability to handle complex relationships and adversarial examples.

Method	Accuracy
TF-IDF + SVM	80%
TF-IDF + LR	80%
TF-IDF + RF	89%
Word2Vec + NBLCR + MLP	86%
N-BEATS	97%

TABLE I
ACCURACY COMPARISON

Fig 2 compare models with precision, recall and F1-Score

- **Precision**:
  - Traditional models (SVM, LR, RF) achieve reasonable precision for both classes but struggle with offensive text due to imbalanced feature importance.
  - Word2Vec + NBLCN + MLP further improves precision by dynamically weighting word vectors, highlighting its robustness.
- **Recall**:
  - Traditional models struggle to recall offensive text (e.g., RF recall = 69% for class 1), indicating a high false negative rate.
  - Beat achieves near-perfect recall, significantly reducing false negatives.
- **F1-Score**:
  - Word2Vec + NBLCN + MLP maintains balanced performance across folds.

B. Adversarial Text Analysis

The Beat model demonstrates superior performance in handling adversarial cases, such as:

- **Example**: *"You are fucking beautiful"*.
  - Beat correctly identifies this as non-offensive with a probability of 0.0056, showing its ability to understand semantic context rather than relying on individual offensive words.

  - Other models (SVM, LR, RF, Word2Vec) classify it as offensive with high probability (e.g., 0.9918), revealing their over-reliance on specific trigger words.

Insights:

- Beat's architecture likely excels due to dynamic embeddings and its focus on the semantic relationships between words.
- Traditional models and even Word2Vec + NBLCN fail to generalize in such cases because they lack dynamic adjustment for word importance.

V. Conclusion

This research successfully explored and implemented multiple models and techniques to address the challenging task of cyberbullying detection. Through an iterative approach, we demonstrated how different networks and methodologies can improve classification accuracy and robustness.

Research Achievements

Our study began with traditional machine learning models, such as SVM, Logistic Regression, and Random Forest, combined with TF-IDF for feature representation. While these approaches provided a solid baseline, they were limited by their inability to capture semantic relationships.

To address these shortcomings, we integrated modern techniques:

- **Word2Vec + NBLCN + MLP**: Improved the representation of sentence semantics, achieving an average accuracy of 86%.
- **N-BEATS**: Introduced dynamic embeddings to adaptively capture context, boosting accuracy to 97%.
- **Adversarial Training**: Enhanced the model's ability to handle edge cases by focusing on semantics rather than specific offensive keywords.

This iterative process demonstrates the critical role of model architecture and training strategy in improving performance. The N-BEATS model, in particular, excelled at handling complex relationships and adversarial cases, significantly outperforming other methods.

Research Limitations

Despite these achievements, certain limitations remain:

- **Sensitivity to Noise**: While adversarial training improved robustness, models may still struggle with noisy or incomplete data.
- **Static Model Outputs**: Our models rely on fixed embedding techniques, which might limit their adaptability to evolving language patterns and emerging slang.
- The study primarily focused on binary classification (offensive vs. non-offensive), leaving more nuanced multi-class categorization (e.g., hate speech, trolling) as a potential area for future research.

Future Directions

Building on this research, several directions can be pursued: Explore advanced transformer-based architectures like GPT and RoBERTa, enhance explainability with techniques such as SHAP or attention visualization, and combine outputs from multiple models using voting or stacking methods to improve robustness, accuracy, and interpretability.

In conclusion, this research highlights the importance of combining traditional methods with modern deep learning techniques to advance the field of cyberbullying detection. By iteratively refining our approach, we achieved a robust and highly accurate model, paving the way for further innovations in this domain.

References

[1] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*

[2] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692.*

[3] Oreshkin, B. N., Carpov, D., Chapados, N., Bengio, Y. (2020). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437.*

[4] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781.*

[5] Xu, J. M., Jun, K. S., Zhu, X., Bellmore, A. (2012). Learning from Bullying Traces in Social Media. In *Proceedings of the 2012 conference of the North American chapter of the Association for Computational Linguistics: Human language technologies* (pp. 656-666).

[6] Ng, A. Y. (2004). Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning* (p. 78).

[7] Mikolov, T., Chen, K., Corrado, G., Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781.*

[8] Pennington, J., Socher, R., Manning, C. D. (2014). GloVe: Global vectors for word representation. *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

[9] De Boom, C., et al. (2016). Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters.*

[10] Torki, M. (2018). A Document Descriptor using Covariance of Word Vectors. *ACL.*

[11] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).*

[12] Wang, S., Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and text classification. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.*

[13] Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, 36(2), 111–147.*