# Semi-Supervised Sentiment Classification via FastText and Gradient Boosting

Alif Ilham Madani
ECE at Cornell Tech
aim57@cornell.edu

## Abstract

*This project develops a sentiment classification model for a 5-class movie review dataset where a significant portion of the training data (58%) is unlabeled. We implement a sequential, five-step optimization pipeline, using validation accuracy to select the best configuration at each stage. The pipeline systematically evaluates: (1) text processing (raw vs. preprocessed), (2) text embeddings (BoW, N-grams, TF-IDF, GloVe, and FastText), (3) baseline models (Logistic Regression, SVM, and XGBoost) on labeled data, (4) label augmentation strategies (Label Propagation vs. supervised pseudo-labeling) to leverage unlabeled data, and (5) a final model retraining on the complete augmented dataset. Our experiments show that raw text, which preserves sentiment cues like emojis, is superior to preprocessed text. FastText (300 dimensions, 20 epochs) significantly outperforms other embeddings (0.9160 validation accuracy). An XGBoost classifier (max depth 7) proves most effective for the high-dimensional data (0.9334 validation accuracy). Finally, using this XGBoost model to generate pseudo-labels for the unlabeled data and retraining on the complete dataset yields a final* ***test accuracy of 0.9350****.*

## 1. Methods

The dataset consists of movie reviews with sentiment labels 0-4. There are 59706 train data points ( 58% are unlabeled), 23256 validation data points, and 23257 test data points.

The comparison methods used in this project are as follows:

1. Text processing
2. Embedding
3. Original label training
4. Label augmentation
5. Final training

The best model in each step is used for the next step. Accuracy is used the metric since the dataset has balanced datapoints for each class. The validation data is used to evaluate the model performance.

### 1.1. Text processing

In the first step, we compare models trained on raw texts and preprocessed texts. The model used is a logistic regression model class with L2 regularization ( $\alpha = 10^{-5}$) and random_state=42 for each experiment. Here, $\alpha$ is the L2 regularization term constant. In the raw texts, null entries are being replaced by empty strings. On the other hand, the preprocessing steps include removing special characters, tokenizing the sentences, lemmatizing the words, and converting the words back into sentences.

We use raw texts because they contain more information, including emojis represented in the sentences, that are being removed in the preprocessing steps. However, we want to compare the preprocessed texts, as they should contain more relevant words that contain more meaningful context related to sentiment analysis. In this step, we use a BoW embedding with 100 dimensions, and only use the labeled data as the training data.

### 1.2. Embedding comparison

After choosing the best text processing, we compare a variety of embeddings, including Bag-of-Words (BoW), N-grams, TF-IDF, GloVe, and FastText. We use scikit-learn package [4] to implement BoW, N-grams and TF-IDF and they have similar idea to get the vector representations of texts from word counts. Another approach we use is GloVe [2, 5] which has pretrained word vectors pretrained on text documents. We also use FastText to train the word representation model [1] which uses N-gram characters. This makes FastText can handle typos in the sentences, also emojis can be represented using this model.

We also use different feature dimensions, N-grams, and epochs to compare the effect for each configuration to the model performance. For each experiment, we use the labeled training data and the same model as described in section 1.1. The embedding configurations are:

1. BoW:
   (a) 100 feature dimensions
   (b) 200 feature dimensions
   (c) 300 feature dimensions
2. N-grams with N=[1,2]:

   (a) 100 feature dimensions
   (b) 200 feature dimensions
   (c) 300 feature dimensions
3. TF-IDF with N-grams=[1,2]:
   (a) 100 feature dimensions
   (b) 200 feature dimensions
   (c) 300 feature dimensions
4. GloVe 2024 Wikipedia + Gigaword 5 (11.9B tokens, 1.2M vocab, uncased, 300 dimensions),
5. FastText:
   (a) 10 epochs and 100 dimensions
   (b) 10 epochs and 200 dimensions
   (c) 10 epochs and 300 dimensions
   (d) 20 epochs and 300 dimensions
   (e) 30 epochs and 300 dimensions

### 1.3. Original label training

After choosing the best embedding, we compare different types of models trained using the original labeled data. We use logistic regression, support vector machines (SVM), and XGBoost model classes for comparison. The logistic regression and SVM are implemented using scikit-learn [4]. Another model class, XGBoost [3] is used as a comparison because it uses a gradient boosting method, combining different decision trees which works well on high-dimensional features. The validation data is used to compare performances across models. The following models and configurations are used for comparison:

1. Logistic Regression with L2 regularization constant=$10^{-5}$
   (a) 1000 maximum iterations
   (b) 2000 maximum iterations
2. SVM with L2 regularization constant=$10^{-5}$
   (a) 1000 maximum iterations
   (b) 2000 maximum iterations
3. XGBoost with softmax objective, regularization $\lambda = 10^{-5}$
   (a) 5 maximum depth
   (b) 6 maximum depth
   (c) 7 maximum depth
   (d) 9 maximum depth

### 1.4. Label augmentation

To increase the model's performance, we incorporate the remaining unlabeled training data. First, we use the best model trained on the original labeled training data. Then, we use the same model to predict the labels. The model classes used are:

1. Semi-supervised learning: Label propagation with kNN kernel
2. Supervised learning: XGBoost with oftmax objective, regularization $\lambda = 10^{-5}$, and 7 maximum depth.

We use the label propagation algorithm to ensure the structure of labeled data and propagate the labels in high-density areas [6]. We also use the supervised learning method with the assumption that it generalizes the prediction on the unlabeled data.

| Configuration | Train Accuracy | Validation Accuracy |
|---|---|---|
| **Raw** | **0.4627** | **0.6257** |
| Preprocessed | 0.4559 | 0.6120 |

Table 1. Text processing results

| Configuration | Train Accuracy | Validation Accuracy |
|---|---|---|
| 1a | 0.4628 | 0.6257 |
| 1b | 0.5282 | 0.7065 |
| 1c | 0.5721 | 0.7406 |
| 2a | 0.4723 | 0.6021 |
| 2b | 0.5313 | 0.6964 |
| 2c | 0.5741 | 0.7380 |
| 3a | 0.4650 | 0.5774 |
| 3b | 0.5258 | 0.6827 |
| 3c | 0.5764 | 0.7171 |
| 4 | 0.6731 | 0.7633 |
| 5a | 0.8055 | 0.9066 |
| 5b | 0.8172 | 0.9116 |
| 5c | 0.8204 | 0.9125 |
| **5d** | **0.8301** | **0.9160** |
| 5e | 0.8295 | 0.9141 |

Table 2. Embedding comparison results

### 1.5. Final training

After having augmented the label, we add the augmented labels to the training data. Then, we retrain the model classes with the full training data consisting of real labels and pseudo labels.

## 2. Results

For each step outlined in section 1, a series of experiments is run to better compare the effect of each configuration.

### 2.1. Text processing

The configurations are stated in section 1.2 with the results stated in table 1.

### 2.2. Embedding comparison

Based on the previous section, the best processing is raw texts which we used in this step. Then, we use the configurations in section 1.2 with the results stated in table 2:

### 2.3. Original label training

Based on the previous section, the best embedding is FastText with 20 epochs and 300 dimensions which we used in

| Configuration | Train Accuracy | Validation Accuracy |
|---|---|---|
| 1a | 0.8250 | 0.9124 |
| 1b | 0.8250 | 0.9124 |
| 2a | 0.8395 | 0.9216 |
| 2b | 0.8395 | 0.9216 |
| 3a | 0.9957 | 0.9295 |
| 3b | 0.9974 | 0.9304 |
| **3c** | **0.9974** | **0.9335** |
| 3d | 0.9957 | 0.9295 |

Table 3. Original label training results

| Configuration | Train Accuracy | Validation Accuracy |
|---|---|---|
| 1 | 0.8859 | 0.9225 |
| **2** | **0.9974** | **0.9335** |

Table 4. Label augmentation results

| Train Accuracy | Validation Accuracy | Test Accuracy |
|---|---|---|
| **0.9989** | **0.9345** | **0.9350** |

Table 5. Final training results

this step. Then, we use the configurations stated in section 1.3 with the results stated in table 3.

### 2.4. Label augmentation

Based on the previous section, the best model is XGBoost with softmax objective, regularization $\lambda = 10^{-5}$, and 7 maximum depth. We then use the configurations in section 1.4.

### 2.5. Final training

Based on the previous section, the best model is XGBoost with softmax objective, regularization $\lambda = 10^{-5}$, and 7 maximum depth. The model is then retrained with the full training data consisting of real and pseudo labels. The accuracy results are presented in Table 5 with the test accuracy taken from Kaggle.

## 3. Discussion

In the text processing step, we found that the raw texts produce higher accuracy compared to preprocessed text when using BoW as the embedding. This can happen because there is information loss when we process the texts, such as removing emojis. In this task, emojis can convey sentiment which is crucial to classify the sentence. Removing punctuation such as exclamation marks also can lead to making the model lose context of the sentences. In this case, using raw or minimal processing is better.

Another important step is choosing the embedding because embedding represents the words and sentences that will be used later. Wrong representations might lead to poor performance in downstream tasks. For example, when we compare the number of feature dimensions in each embedding, increasing the feature dimensions lead to better performance. Using more contextual embedding also increases performance as we change from count-based embedding to GloVe.

FastText shows better performance than GloVe because it uses N-gram characters to train the word representations. This allows the embedding to be typo-resistant since slight differences in word characters can be captured. FastText also allows capturing rare words that come from two common words. For example, ”book” and ”shelf” are common, but ”bookshelf” is less common. Using N-gram characters, allows FastText to capture this relationship. Increasing the dimensions also increases the accuracy up to a certain point.

In the original label training step, we found that linear models such as logistic regression and SVM failed to fit the data. This is shown by the training accuracy, which is much lower than the validation accuracy. This is proven by no performance improvement when increasing the max iterations parameter. This indicates more complex model classes are needed to fit the data.

We used XGBoost because it performs well with high-dimensional data. Using boosting also helps to increase the performance, since the model aggregates the prediction from various decision trees. Increasing the maximum depth also helps the model to capture more relationships in the data. This increases the model performance. However, after reaching a certain point, increasing the maximum depth decreases the performance. This is because the model can overfit to the training data, so it suffers from bias when tested on the validation data.

In the label augmentation step, we compare label propagation and XGBoost algorithms. The objective is to create pseudo labels after having learned the training data. The XGBoost model performs better since it directly learns the data structure based on the real labels only. When the label propagation model propagates the label, the predictions on the real labels might get affected to account for the generalization of unlabeled data.

## 4. Conclusion

In this paper, we successfully constructed a high-performance sentiment analysis classifier by following a systematic, five-stage optimization pipeline. This approach was particularly effective for a dataset with a large volume of unlabeled data.

Our key findings demonstrate the importance of methodical selection at each step:

1. Text Processing: We found that using raw text inputs yielded better performance than preprocessed text, indicating that punctuation and emojis contain valuable sentiment information that is lost during cleaning.

2. Embeddings: The choice of embedding was critical. FastText (300 dimensions, 20 epochs) provided a substantial performance increase over all other methods, including GloVe, likely due to its ability to handle typos and sub-word information.
3. Model Selection: The XGBoost classifier (max depth 7) was essential for capturing the complex, high-dimensional patterns in the FastText embeddings, a task where linear models like Logistic Regression and SVM clearly underfit.
4. Unlabeled Data: We effectively leveraged the 58% unlabeled data by using our best XGBoost model to generate pseudo-labels. This supervised augmentation approach outperformed the semi-supervised Label Propagation method.

By retraining the optimized XGBoost model on the final, fully augmented dataset (combining original labels and pseudo-labels), we achieved a final test accuracy of 0.9349. This result validates that a sequential pipeline combined with supervised pseudo-labeling is a highly effective strategy for this sentiment classification task.

## References


[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. 1

[2] Riley Carlson, John Bauer, and Christopher D. Manning. A new pair of gloves, 2025. 1

[3] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, Jiaming Yuan, and David Cortes. *xgboost: Extreme Gradient Boosting*, 2025. R package version 3.2.0.0. 2

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 1, 2

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 1

[6] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. 2