Decision Trees vs Neural Networks for Classification of Emotions from Text

Ettore Ricci

October 2, 2024

1 Introduction

In today's digital world, the ability to interpret emotions from written text has gained importance for a variety of applications, including customer feedback analysis and mental health monitoring. Emotion classification, a challenging task within natural language processing, focuses on identifying the emotional tone within text.

This work compares the performance of three different models for emotion classification: **decision trees**, **random forests** and **neural networks**. The models were trained on a dataset of online messages labeled with 7 different emotions: Happiness, Sadness, Anger, Worry, Love, Surprise and Neutral.

2 Data

The dataset used was a mix of the CrowdFlower dataset[1] and the Emotions dataset[2]. The CrowdFlower dataset contains around 40,000 tweets labeled with 13 different emotions. The Emotions dataset contains around 400,000 tweets labeled with 6 different emotions.

2.1 Exploratory data analysis and preparation

Some emotions that are present in the CrowdFlower dataset were remapped because they were not present in the Emotions dataset. The emotions and relative mappings are shown in Table 1.

Emotion	Mapping
Happiness	
Sadness	
Anger	
Worry	
Love	
Surprise	
Neutral	
Fun	Happiness
Relief	Happiness
Hate	Anger
Empty	Neutral
Enthusiasm	Happiness
Boredom	Neutral

Table 1: Emotion mappings, when the mapping is empty the emotion was not remapped.

Some instances were removed because they were duplicates. There were 23,163 duplicates and the dataset was reduced to 433,646 instances.

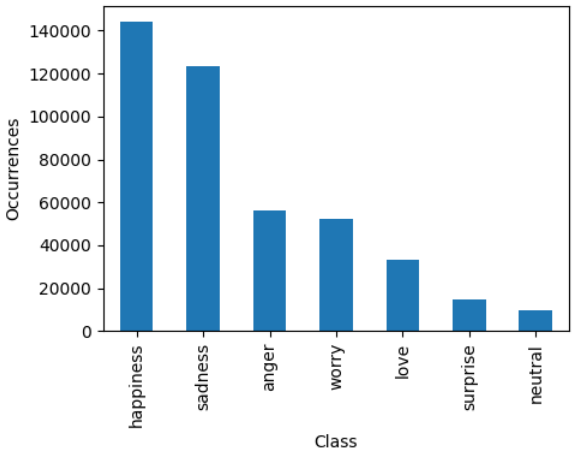

Figure 1: Class distribution of the full dataset

As shown in Figure 1, the dataset is imbalanced.

The most common emotion is Happiness, and the least common is Neutral. This is because the larger dataset (Emotions) does not contain the Neutral class.

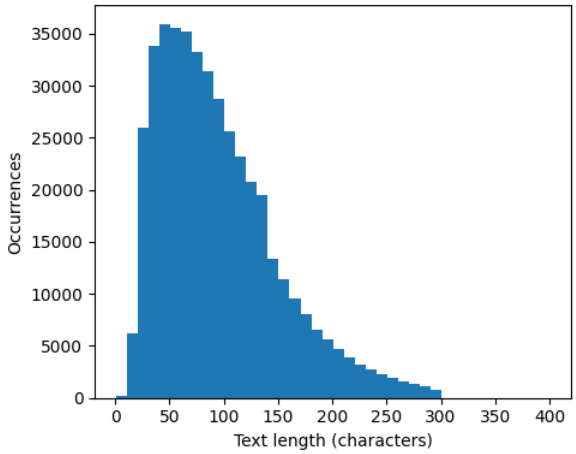

Figure 2: Text length distribution of the full dataset

The dataset was split into a training set and a test set with a 80% - 20% ratio.

2.2 Preprocessing

Multiple kinds of preprocessing were tested:

- *Bag of words (BoW)*
- *GloVe embeddings (GE)*

For the BoW preprocessing, the text was first cleaned with a regular expression that removes urls, mentions and symbols. Then the text was tokenized and stemmed with the Snowball stemmer. **TFIDF**, **count** and **binary** encodings were tested for the BoW preprocessing. The GE preprocessing was done by applying the same cleaning regular expression but without stemming. The vectors used were the 6B 50d vectors from GloVe[3]. GE preprocessing consists in word embeddings with pretrained vectors.

3 Models

Three different categories of models were considered:

- *Decision Trees*
- *Random Forests*
- *Neural Networks*
  - *Feedforward Networks*
  - *LSTM Networks*
  - *Transformer Networks*

Decision trees and **random forests** both use the BoW preprocessing. feedforward **neural networks** use the BoW preprocessing with **TFIDF** encoding. LSTM Networks and Transformer Networks use the GE preprocessing.

3.1 Imbalanced Dataset

To handle the imbalanced dataset, a class weight was assigned to each class. The weight for a class c was calculated as $W_c = \frac{|D|}{|C||D_c|}$ where $|D|$ is the size of the dataset, $|C|$ is the number of classes and $|D_c|$ is the number of samples in class c. This weight was used in the criterion of the **decision trees** and **random forests** models, and were used for resampling the training dataset (with replacement) in the **neural networks** models.

3.2 Model Selection

To find a good hyperparameter configuration for each model, a random search was performed. The search was done with 10 iterations for each model (**decision trees**, **random forests** and **neural networks**). Each configuration was evaluated with a 6-fold cross-validation on the training set. Two subsequent random searches were performed, the second one narrowing the search space around the best configuration found in the first search. The search spaces for each model are shown in Table 2, Table 3 and Table 4 for the first search, and in Table 5, Table 6 and Table 7 for the second search. Neural Networks used the **early stopping** technique to avoid overfitting and speed up the training process.

Hyperparameter	Search Space
`preprocessor`	{tfidf, binary, count}
`criterion`	{gini, log_loss}
`splitter`	{best}
`max_depth`	{10, 100, None}
`min_impurity_decrease`	{0.01, 0.0001, 1e-06, 1e-08, 0}
`class_weight`	{balanced}

Table 2: Decision Trees hyperparameter search space for the first search.

Hyperparameter	Search Space
n_estimators	{10, 50, 100}
preprocessor	{tfidf, binary, count}
criterion	{gini, log_loss}
max_depth	{10, 100, None}
min_impurity_decrease	{0.01, 0.0001, 1e-06, 0}
n_jobs	{-1}
class_weight	{balanced}

Table 3: Random Forest hyperparameter search space for the first search.

Hyperparameter	Search Space
network	{ff_tfidf, lstm_glove, tnn_glove}
base_size	{8, 16, 32}
depth	{1, 2, 3, 4}
patience	{2}
epochs	{20}
dropout	{0.5}
batchnorm	{True, False}
batch_size	{32}
lr	{0.01, 0.001}
optimizer	{adam}

Table 4: Neural Networks hyperparameter search space for the first search. While using the `ff_tfidf` network, if `batchnorm` is set to `True`, the `dropout` hyperparameter is set to 0. When using the `lstm_embeddings` and `lstm_glove` networks, the `batchnorm` hyperparameter is set to `False`.

Hyperparameter	Search Space
criterion	{gini, log_loss}
splitter	{best}
max_depth	{1000, None}
min_impurity_decrease	{1e-05, 1e-06, 1e-07}
class_weight	{balanced}

Table 5: Decision Trees hyperparameter search space for the second search.

Hyperparameter	Search Space
n_estimators	{100, 125, 150}
criterion	{gini, log_loss}
max_depth	{1000, None}
min_impurity_decrease	{1e-05, 1e-06, 1e-07}
n_jobs	{-1}
class_weight	{balanced}

Table 6: Random Forest hyperparameter search space for the second search.

Hyperparameter	Search Space
network	{lstm_glove}
base_size	{24, 32, 48}
depth	{2, 3, 4}
epochs	{20}
patience	{2}
dropout	{0.4, 0.5, 0.6}
batchnorm	{False}
batch_size	{32}
lr	{0.001}
optimizer	{adam}

Table 7: Neural Networks hyperparameter search space for the second search. The same rules described in Table 4 apply.

4 Results

The models were compared using their mean F1 (normalized by class) in the 6-folds over the validation set. We can se the results for each model in Figure 3.

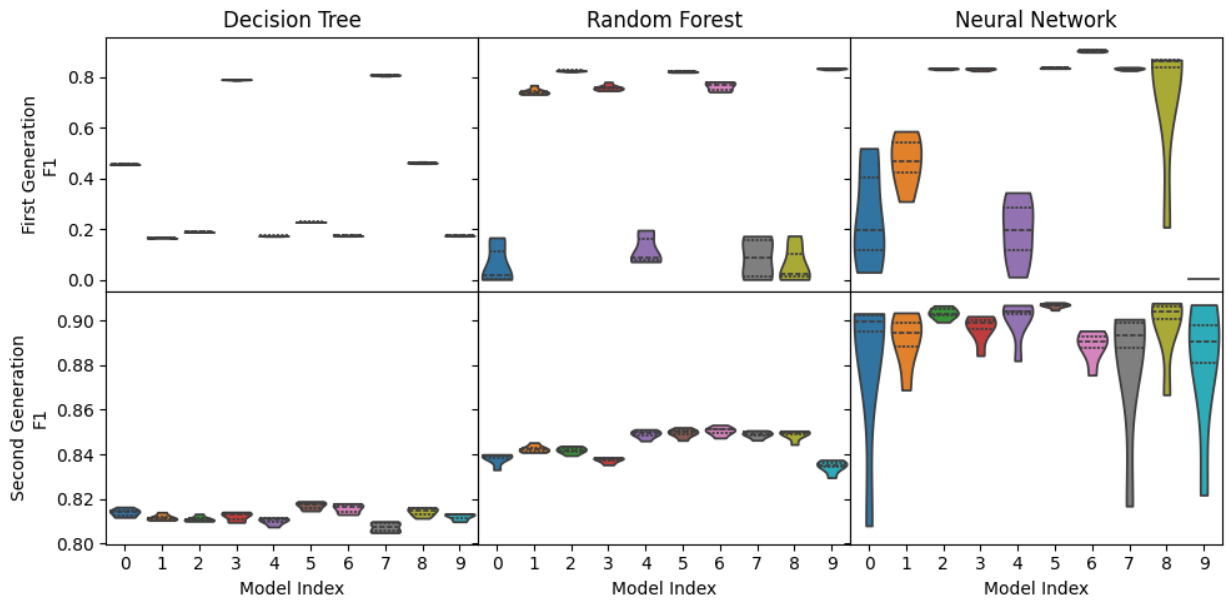

Figure 3: F1 distribution of each model configuration tested during the model selection process.

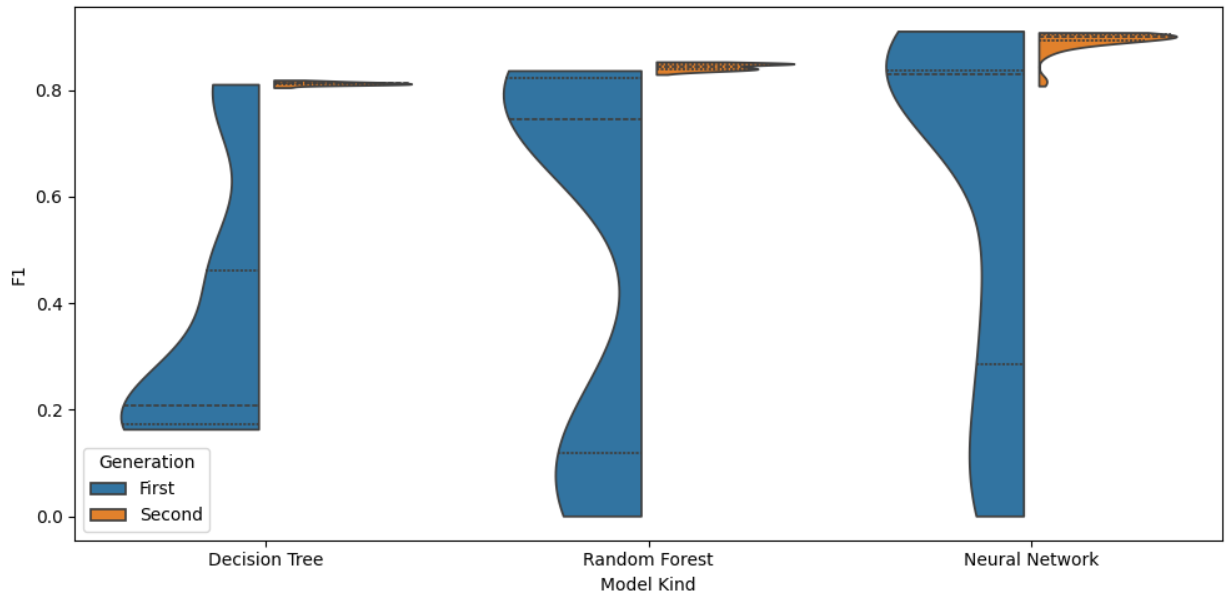

Figure 4: F1 distribution of each model category.

Among the models tested in this work, the best one was the **neural network** followed by the **random forest** and then the **decision treee**. The best models for each kind are shown in Table 8 and their respective hyperparameters are shown in Table 9, Table 10 and Table 11.

Model Name	Mean F1
`decision_tree-G1-5`	81.71%
`random_forest-G1-6`	85.05%
`neural_network-G1-5`	90.68%

Table 8: Best models for each kind

Hyperparameter	Value
network	lstm_glove
base_size	32
depth	3
epochs	20
patience	2
dropout	0.5
batchnorm	False
batch_size	32
lr	0.001
optimizer	adam

Table 9: Hyperparameters used for model neural_network-G1-5.

Hyperparameter	Value
n_estimators	150
preprocessor	count
criterion	gini
max_depth	None
min_impurity_decrease	1e-07
n_jobs	-1
class_weight	balanced

Table 10: Hyperparameters used for model random_forest-G1-6.

Hyperparameter	Value
preprocessor	tfidf
criterion	gini
splitter	best
max_depth	1000
min_impurity_decrease	1e-06
class_weight	balanced

Table 11: Hyperparameters used for model decision_tree-G1-5.

The best model was selected based on the average F1 obtained in the cross-validation. The F1 distribution of this model was compared to the other models with a Wilcoxon signed-rank test, some models were not significantly different ($p > 0.05$) from the best model and they are listed in Table 12.

Model	Mean F1
neural_network-G1-5	90.68%
neural_network-G0-6	90.54%
neural_network-G1-2	90.33%
neural_network-G1-4	90.06%
neural_network-G1-8	89.82%

Table 12: Best model (first row) and models not significantly different from the best model.

As we can see in Figure 4, the random forest is the most susceptible to hyperparameter changes, as it has the widest F1 distribution, while the decision tree is the least susceptible but still the worse performing, with the smallest F1 distribution. The neural network is in the middle, with a distribution skewed to higher F1 values.

Metric	Value
Precision	94.23%
Recall	89.33%
F1	91.06%

Table 13: Metrics relative to the best model neural_network-G1-5 evaluated on the test set.

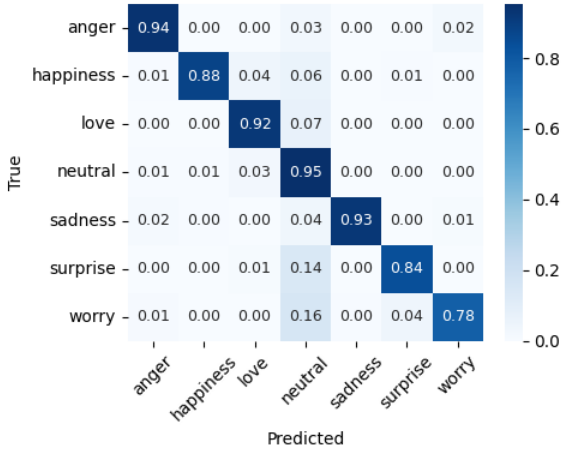

Figure 5: Confusion matrix of the best model.

5 Conclusion

In this work three different kinds of machine learning algorithms were compared in the task of emotion classification from text. Random forest has proven to be the best performing model among the ones tested while the decision tree was the most stable one. The neural network was the worst performing model, but it was the one that had the most room for improvement. Unfortunately, the computational resources available were not enough to explore more hyperparameters and configurations for the neural network.

5.1 Future Work

The neural network model has great room for improvement and maybe a more modern architecture like a transformer could have better results. Also exploring more hyperparameters for every model could lead to better results.

References

[1] C. Van Pelt and A. Sorokin, "Designing a scalable crowdsourcing platform," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 765–766, 2012.

[2] N. Elgiriyewithana, "Emotions." https://www.kaggle.com/dsv/7563141, 2024.

[3] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.