Nethra Middela

# Building and Evaluating an Emotion Recognition Model

## Purpose

As part of the Independent Study and Mentorship program at Heritage High School, I was researching and exploring a field of my choice: Artificial Intelligence (AI). The program's objective was to gain insight and experience into a possible future career. During the first semester of the program, I completed individual research on Artificial Intelligence to see if it was really a field I wanted to pursue. Toward the end of the first semester, I secured a mentorship with Ms. Namratha Urs, a Ph.D. student and teaching assistant at the University of North Texas, who specializes in applied machine learning and natural language processing to help me with my research and learning. I explained my vision of what I wanted to learn, and after a few rounds of discussion and exploration, we decided on how to pursue my original work.

For my original work, I learned how to build an emotion recognition model from scratch to attain a deeper understanding of working with natural language processing in artificial intelligence.

Artificial intelligence is the ability of a machine to perform human-like tasks by analyzing patterns in billions of data, then predicting the outcome based on that analysis. There are several sub-topics in AI, one of them being Natural Language Processing (NLP). NLP is the ability of machines to understand natural language in a written or spoken form. Some examples of NLP are speech recognition from Google's Google Home, Amazon's Alexa, and iPhone's Siri; text-to-speech readers that would read out words, a search engine that picks up keywords and returns appropriate results, and predictive text like Grammarly that would complete and edit sentences.

## Idea Brainstorming

The previous goal was to complete a sentiment analysis model, which is a model that would predict the intensity of sentiment, or emotion, in a statement. If the sentiment was geared towards the emotion of sadness, then a sentence like *I was sad about the car crash* would return a value of lesser sentiment than *My heart was heavy from learning about the tragic accident*. However, after further discussion with Ms. Urs, I was envisioning more of an emotion recognition model, where, given a certain piece of text, the model can predict what emotion the text most displays.

Because what I wanted to complete was more complex than time permitted, the main goal for me was to build and evaluate how effectively the emotion recognition model would predict the

emotion if given an unseen piece of text, because, in the professional world, a dysfunctional model is useless if it does not predict accurately.

The use cases I envisioned for the emotion recognition model was to read and evaluate a student's essay, blog, email, or other writing piece to gauge their interest, involvement, and overall feeling towards the subject. I believe a lot of a student's thinking and mindset can be revealed through how they write, so, in my goal to further modern education by incorporating Artificial Intelligence, this model will pick up the minute details that a normal reader would miss by analyzing a student's work. For example, a very engaged, interested, and involved student may be caught using more subject-relevant, articulated vocabulary, while a less engaged student may write more generally with less enthusiastic diction choice. Of course, the performance of this use case relies on more than just a powerful NLP emotion recognition model, but also human psychology, cognitive science, and relies heavily on the purpose of the writing piece itself.

**Resources**

The libraries used include a combination of Pandas, Scikit-Learn, Natural Language Toolkit, and NumPy. I used the Google Colab notebook to store and write all the code. The dataset is the emotion dataset from Hugging Face API.

**Machine learning workflow pipeline**

To build a machine learning workflow pipeline, I had to go through the certain steps of collecting data, preprocessing, feature extraction, training and validation, and evaluation.

**Dataset**

I used the emotion dataset from Hugging Face API. The data was made up of twitter tweets, and came pre-labeled with six emotions: joy, sadness, anger, fear, love, and surprise, represented with numbers 1, 2, 3, 4, 5, and 6 respectively. The data was also split into a 80% training set, 10% validation set, and 10% testing set. The training data is used to train the model, the validation set is used to adjust the hyperparameters of the learning algorithm to make the model as accurate as possible for the best outcome, and the testing set is used to evaluate the performance of the model after every possible adjustment. Each piece of data would have two elements to it: the text data, which is the alphabetical text itself, and the label data, which is the number representing the emotion. I downloaded and loaded the dataset into Google Colab.

**Preprocessing**

First, I preprocessed the training text data. Preprocessing is cleaning up the text data to allow the machine to understand it more easily. Which means, I needed to remove all capitalization, punctuation, numbers, and URLs. So, a complex tweet would be reduced down and put into a vocabulary list. For example, “I love listening to Gorgeous Girl’s 5th Album!” becomes [“i”, “love”, “listening”, “to”, “gorgeous”, “girls”, “th”, “album”]. The emotion text dataset already came preprocessed, but I reapplied it just in case it missed out on any of the specifications I wanted to be removed.

I also preprocessed the training label data, which is much simpler as the label data is already represented in a numerical form (1 to 6 for joy to surprise).

**Feature Extraction**

Next, I extracted the features from the dataset. Feature extraction means to find features, also known as patterns, in the data. There were two methods I used to do this: the Bag of Words method and the Term Frequency-Inverse Document Frequency (or TF-IDF) methods. From here on, I started building two different pipelines, one for each method, just for organization purposes. However, most of the process after the feature extraction is the same.

**Bag of Words**

In the Bag of Words (BoW) pipeline, after preprocessed text data was put into a vocabulary list, each word was assigned a word embedding/vector, which is a numerical representation of that word. Then, the machine went over the whole vocabulary list. In nonbinary BoW, which is the method I used in my pipeline, the machine counts and returns the number of times each word appears in the dataset. In binary BoW, the machine returns a 1 for if the word is present in that sample text data, or 0 if it is not. For example, with the dataset:

- Today is a lovely day.
- Tomorrow is a lovely day.
- Today is the day before tomorrow.
- Tomorrow is the day after today, tomorrow.

The nonbinary vocabulary list includes:

- “a” = 2
- “after” = 1
- “before” = 1
- “day” = 4
- “is” = 4
- “lovely” = 2
- “today” = 3
- “the” = 2
- “tomorrow” = 4

The binary vocabulary list includes:

- “a” = 1
- “after” = 0
- “before” = 0
- “day” = 1
- “is” = 1
- “lovely” = 1
- “today” = 1
- “the” = 1
- “tomorrow” = 1

For a nonbinary vocabulary list, the sentence “Tomorrow is the day before today, tomorrow” is represented by [0, 0, 1, 1, 1 , 0, 1, 1, 2], which is returning the word count.

For a binary vocabulary list, the sentence “Tomorrow is the day before today, tomorrow” is represented by [0, 0, 1, 1, 1 , 0, 1, 1, 1], which is returning the appearance of the word.

**TF-IDF**

In the TF-IDF pipeline, the method runs through the trained text data vocabulary list and sees how often the word appears in that setting. It goes through an equation similar to this:

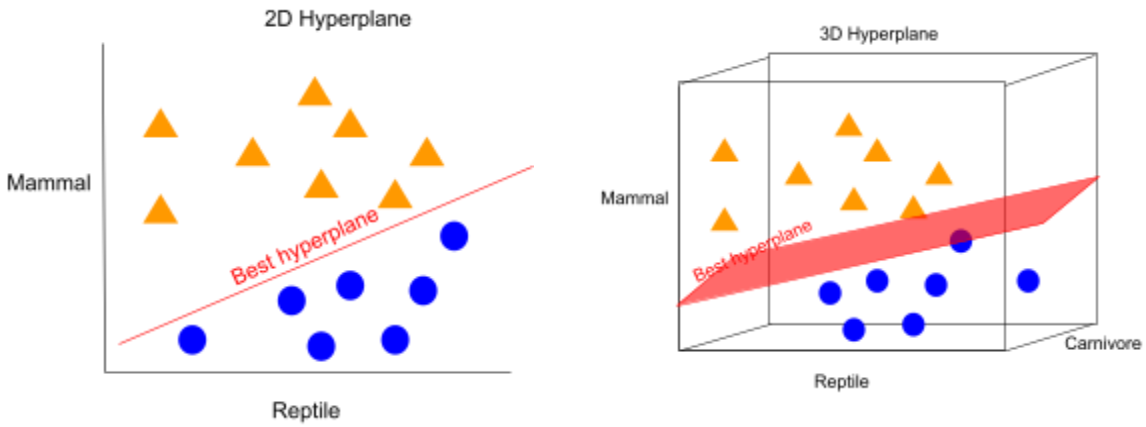
$$TFIDF = \frac{word\ count\ of\ term\ x\ in\ document\ y}{total\ number\ of\ words\ in\ document\ y} \times log\left(\frac{total\ number\ of\ documents}{number\ of\ documents\ including\ the\ term\ x}\right)$$

The words that are less common have a higher TF-IDF than words that are more common, which show the importance of that word in that context.

**Learning Algorithm**

After extracting the features (frequency, appearance, rarity, etc.) of a sample using BoW and TF-IDF, I fitted the trained text and label data into a learning algorithm, called the Support Vector Machine, or an SVM. A learning algorithm helps the machine categorize the input into the most appropriate output. The SVM, in particular, helps the machine create a hyperplane, or a dividing line, between each type of classification, based on the number of features extracted. With two features, SVM makes a 2D hyperplane, with three features, SVM makes a 3D hyperplane.

After training the model on the training data, I repeated preprocessing on the validation text and label data because I need to apply it to the SVM to evaluate the performance of the model. Then, I made the model predict the label validation data from the text validation data.




**Evaluation**

Evaluation is evaluating the performance of the model, and seeing if it works correctly and realistically. There are a few different evaluation metrics that I used for evaluation: accuracy score, precision score, recall score, F1-Score, and overall, the Confusion Matrix. Certain metrics are used for the appropriate scenario.

- Accuracy: $\frac{All\ correct\ guesses}{Total\ guesses}$
- Precision: $\frac{Correct\ Target\ Category\ guesses}{Predicted\ Total\ Target\ Category\ guesses}$
- Recall: $\frac{Correct\ Target\ Category\ guesses}{Actual\ Total\ Target\ Guesses}$
- F1-Score: $\frac{2 \times Precision \times Recall}{Precision + Recall}$

For example, a precision score is used for spam email detection. Precision score marks more items as positive. Precision allows more spam to be marked as non-spam, because it is more important to not lose an important email rather than removing spam email. Recall score works the opposite, marking more negatives. Recall can be used for fraud or illness detection, where it is more important that the fraud/illness is identified rather than marking the scenario as non harmful. F1-Score is used when there needs to be a balance between Precision score and Recall score.

**Adjusting Hyperparameters**

It is not enough to evaluate the model. After evaluation, I learned about implementing a grid search and/or cross validation to adjust the hyperparameters of the SVM model, for better accuracy in its predictions. Grid search is when the model runs through cross validation with each of the possible hyperparameters, then chooses the best combination for the model. Cross

validation is the model returning a score by chunking the validation data into a certain number of kernels, or groups, then using each kernel as a training/validation set to extend and vary the amount of data that the model sees. The score that cross validation returns to be the highest will determine the best hyperparameters through grid search. To illustrate grid search:

**Grid Search**

Assume, three hyperparameters: C, kernel, and degree
Further assume, C = {1.0, x, y} and kernel = {linear, poly, rbf}, degree = {3, 4, 5, 6}

Running the grid search…

- C = 1.0, kernel = linear, degree = 3
- C = 1.0, kernel = poly, degree = 3
- C = 1.0, kernel = poly, degree = 4
- C = 1.0, kernel = poly, degree = 5
- C = 1.0, kernel = poly, degree = 6
- C = 1.0, kernel = rbf, degree = 3

- C = x, kernel = linear, degree = 3
- C = x, kernel = poly, degree = 3
- C = x, kernel = poly, degree = 4
- C = x, kernel = poly, degree = 5
- C = x, kernel = poly, degree = 6
- C = x, kernel = rbf, degree = 3

- C = y, kernel = linear, degree = 3
- C = y, kernel = poly, degree = 3
- C = y, kernel = poly, degree = 4
- C = y, kernel = poly, degree = 5
- C = y, kernel = poly, degree = 6
- C = y, kernel = rbf, degree =3

<u>Algorithm for grid search</u>

```
Loop over every value in C
        Loop over every value in kernel
                Loop over every value in degree
                        …..
                        …..
```

If this combination: C = x, kernel = poly, degree = 4; returns the highest score based on the appropriate evaluation metric, then those are the best hyperparameters to be used on the model.

To illustrate cross validation:

**Cross Validation**

Takes the train set (100 examples)

- Divides the train set into k-splits (kernel is 5 splits); each split would have 20 examples
- Trains and tests (validates) the model 'k' number of times to go through every combination

| | 1 | 2 | 3 | 4 | 5 | | |
|---|---|---|---|---|---|---|---|
| Round 1 | Tr | Tr | Tr | Tr | Val | ; | Tr (80 ex.); Val (20 ex.) |
| Round 2 | Tr | Tr | Tr | Val | Tr | ; | Tr (80 ex.); Val (20 ex.) |
| Round 3 | Tr | Tr | Val | Tr | Tr | ; | Tr (80 ex.); Val (20 ex.) |
| Round 4 | Tr | Val | Tr | Tr | Tr | ; | Tr (80 ex.); Val (20 ex.) |
| Round 5 | Val | Tr | Tr | Tr | Tr | ; | Tr (80 ex.); Val (20 ex.) |

The process is, while grid search is running:

1. On round 1, train the model on folds 1-4 with the according hyperparameters
2. Apply the trained model on the validation fold (fold 5)
3. Evaluate the performance of the model using the metric of choice, and return a score
    a. Repeat steps 1 to 3 for all kernel-number rounds
4. Calculate and return the average of all five scores
5. Go back to grid search, and repeat steps 1 to 4 with the next hyperparameter combination
6. After going through all hyperparameter combinations, the combination to use is the one that returned the highest average score from cross validation.

**Conclusion**

All in all, my model can now be adjusted and grid search and cross validation can be implemented for better performance. The preprocessing specifications can also be improved upon, like removing stop words (like me, I, and, etc.). I have not yet applied the testing data on the model, because this is only supposed to be applied once I believe that the model has reached peak performance, and no other hyperparameter combinations or additions can improve the

model. I will save the testing data for later, after the completion of a few more projects on the model.

As of July 7, 2022, the BoW Pipeline model has:

- SVM Accuracy Score >>> 0.8845
- SVM Precision Score >>> 0.8633392009689936
- SVM Recall Matrix Score >>> 0.8515944522248847
- SVM F1-Score Matrix Score >>> 0.857264148925108

As of July 7, 2022, the TF-IDF Pipeline model has

- SVM Accuracy Score >>> 0.889
- SVM Precision Score >>> 0.8825355792373655
- SVM Recall Matrix Score >>> 0.8366373712717831
- SVM F1-Score Matrix Score >>> 0.85678796718663

The performance of the model varies by evaluation metric and feature extraction methods. Overall, this completes my journey with creating an emotion recognition model and evaluating its performance.

**Works Cited**

Bedi, Gunjit. "Simple Guide to Text Classification(NLP) Using SVM and Naive Bayes with Python." *Medium*, 13 July 2020, https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34.

Brownlee, Jason. "Overfitting and Underfitting With Machine Learning Algorithms." *Machine Learning Mastery*, 20 Mar. 2016, https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

Dataman, Chris. "Use Google DeepMind's Text-to-Speech Natural Voices." *Dataman in AI*, 29 Mar. 2022, https://medium.com/dataman-in-ai/create-your-first-text-to-speech-natural-voices-f71f8f7b5168.

---. "Use Google DeepMind's Text-to-Speech Natural Voices." *Dataman in AI*, 29 Mar. 2022, https://medium.com/dataman-in-ai/create-your-first-text-to-speech-natural-voices-f71f8f7b5168.

"Deeply Moving: Deep Learning for Sentiment Analysis." *Deeply Moving: Deep Learning for Sentiment Analysis*, http://nlp.stanford.edu/sentiment/index.html. Accessed 7 Dec. 2022.

"Evaluating Machine Learning Models." *YouTube*, 3 Feb. 2019, https://www.youtube.com/watch?v=FeKSQy5t_TI.

---. *YouTube*, 3 Feb. 2019, https://www.youtube.com/watch?v=FeKSQy5t_TI.

Gandhi, Rohith. "Introduction to Machine Learning Algorithms: Linear Regression." *Towards Data Science*, 28 May 2018, https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a.

---. "Support Vector Machine — Introduction to Machine Learning Algorithms." *Towards Data Science*, 5 July 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

Nighania, Kartik. "Various Ways to Evaluate a Machine Learning Models Performance." *Towards Data Science*, 30 Jan. 2019, https://towardsdatascience.com/various-ways-to-evaluate-a-machine-learning-models-performance-230449055f15.

Pandey, Parul. "Analysis of the Emotion Data—a Dataset for Emotion Recognition Tasks." *Towards Data Science*, 17 Sept. 2021, https://towardsdatascience.com/analysis-of-the-emotion-data-a-dataset-for-emotion-recognition-tasks-6b8c9a5dfe57.

Pant, Ayush. "Workflow of a Machine Learning Project." *Towards Data Science*, 23 Jan. 2019, https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94.

Randerson112358. "Build A Text To Speech Program Using Python." *Medium*, 9 Apr. 2022, https://randerson112358.medium.com/build-a-text-to-speech-program-using-python-b70de7105383.

---. "Build A Text To Speech Program Using Python." *Medium*, 9 Apr. 2022, https://randerson112358.medium.com/build-a-text-to-speech-program-using-python-b70de7105383.

Saini, Mohit. "Voice Cloning Using Deep Learning." *The Research Nest*, 6 Feb. 2020, https://medium.com/the-research-nest/voice-cloning-using-deep-learning-166f1b8d8595.

Saravia, Elvis, et al. *Https://Aclanthology.Org/D18-1404.Pdf*. National Tsing Hua University, pp. 1–11, https://aclanthology.org/D18-1404.pdf. Accessed 7 July 2022.

Sharma, Gaurav. "Regression Algorithms | 5 Regression Algorithms You Should Know." *Analytics Vidhya*, 26 May 2021, https://www.analyticsvidhya.com/blog/2021/05/5-regression-algorithms-you-should-know-introductory-guide/.

Shetty, Badreesh. "Natural Language Processing(NLP) for Machine Learning." *Towards Data Science*, 24 Nov. 2018, https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b.

---. "Natural Language Processing(NLP) for Machine Learning." *Towards Data Science*, 24 Nov. 2018,

https://towardsdatascience.com/natural-language-processing-nlp-for-machine-learning-d44498845d5b.

Shigabeev, Ilya. "Building Your Own Voice Assistant, Part 1. Text to Speech." *Analytics Vidhya*, 11 Sept. 2020, https://medium.com/analytics-vidhya/building-your-own-voice-assistaint-part-1-text-to-speech-fe76491f9925.

---. "Building Your Own Voice Assistant, Part 1. Text to Speech." *Analytics Vidhya*, 11 Sept. 2020, https://medium.com/analytics-vidhya/building-your-own-voice-assistaint-part-1-text-to-speech-fe76491f9925.

Shreyas, Prajwal. "Sentiment Analysis for Text with Deep Learning." *Analytics Vidhya*, 28 Apr. 2021, https://medium.com/analytics-vidhya/sentiment-analysis-for-text-with-deep-learning-2f0a0c6472b5.

---. "Sentiment Analysis for Text with Deep Learning." *Analytics Vidhya*, 28 Apr. 2021, https://medium.com/analytics-vidhya/sentiment-analysis-for-text-with-deep-learning-2f0a0c6472b5.

"Support Vector Machines (SVM) Algorithm Explained." *MonkeyLearn Blog*, 22 June 2017, https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/.

Tan, Edwin. "How To Train A Deep Learning Sentiment Analysis Model." *Towards Data Science*, 28 Jan. 2022, https://towardsdatascience.com/how-to-train-a-deep-learning-sentiment-analysis-model-4716c946c2ea.

---. "How To Train A Deep Learning Sentiment Analysis Model." *Towards Data Science*, 28 Jan. 2022, https://towardsdatascience.com/how-to-train-a-deep-learning-sentiment-analysis-model-4716c946c2ea.

Tripathi, Himanshu. "What Is Balance And Imbalance Dataset?" *Analytics Vidhya*, 18 Jan. 2021, https://medium.com/analytics-vidhya/what-is-balance-and-imbalance-dataset-89e8d7f46bc5.

Wong, Jason. "Natural Language Processing Workflow." *Towards Data Science*, 4 Dec. 2020, https://towardsdatascience.com/natural-language-processing-workflow-1dddf3a48ab5.