ACM Word Template for SIG Site

Reilly Farrell
Third Year BSCS
Northeastern University
farrell.re@northeastern.edu

Christopher He
Third Year BSCS
Northeastern University
he.chr@northeastern.edu

ARTIST'S STATEMENT

As avid machine learning coders who enjoy art, it has been hard to find new art that we can connect with, that matches the way we feel. Throughout our lives, finding art has been done through Google search, where the results are vague and generic, and our emotions are unable to be visualized. We have been inspired by the powerful techniques of Natural Language processing, and have sought to explore the avenues of emotional expression through language and imagery. The images we generate are well-established, highly renowned pieces of art that have been observed to evoke emotion.

ABSTRACT

In this paper, we describe the process for generating paintings using natural language processing. Generation will be based on sentiment analysis done on user input, pulled from a collection of 4000+ famous paintings, each annotated with a set of emotions that the piece brings to mind. We wanted to combine our knowledge of natural language processing techniques with a real-world, interactive visual application that allows users to describe objects and emotions and bring that before their eyes.

1. INTRODUCTION

We used techniques such as sentiment analysis and text classification, gained from the course, to produce visualizations with sentiment that matched user input. The main flow of the program allows users to enter any text, have its nouns identified along with the sentiment, and return a famous painting with similar objects and sentiment. The inspiration behind this is that it allows users to conjure tangible representations of these expressions in a more interesting way. Text Classification is a category of NLP tasks with many real-world applications, such as detecting spam, fraud, and bots. For our purposes,Text Classification was implemented using the scikit-learn Naive Bayes library and the Keras Neural Network library to label strings of text with certain emotions that are likely to be evoked.

2. DATASETS

The first 2 data sets utilized were for creating word embedding models. We utilized word2vec's brown model and word2vec's news data set models to produce our embeddings. The first database, brown, comes from Brown University and is a collection of text samples of American English. There are 500 samples and 15 genres such as Learned, Humor, Press, etc.. This dataset will be used to create a word2vec Continuous Bag of Words model. The other word2vec model trained on Google News articles also utilizes a continuous bag of words. The vectors are part of Google's pretrained News[3] dataset which includes over 100 billion words. The next datasets were used for our sentiment analysis models. We first began testing an implementation that was trained on a corpus[2]consisting of 'I' statements such as "I am feeling sad today", "I am having the time of my life", etc… Upon evaluating a trained Keras neural network we discovered that the results were lackluster and limited. Most of the data was negative, there was not a diverse range of emotions, and each sentence had only a single corresponding label. Mapping a single emotion to a painting with multiple labels produced unimpressive results and additionally, the model clearly favored joy thus it was thrown out in favor of a more robust dataset. The Go Emotions[1] dataset consists of 27 emotions, 58k entries, and are mapped to multiple labels. These entries were taken from reddit comments and better align with what we would expect users of our software to enter. We trained our Multinomial Naive Bayes and Neural Network models using this dataset. With two models, we were able to compare the results and performance between them. Lastly, we used the WikiArtEmotions[7]paintings dataset, which consists of over 4000 paintings, each with a set of emotion labels that the painting is thought to evoke. This dataset provided the paintings we used for generation based on sentiment.

3. MODELS

Our project consists of 6 NLP models. We utilized 2 models for each NLP task in our project so we could have a larger dimension of comparison. The first use includes a part of speech tagger. The two models/libraries we utilized for this task were NLTK's pos tagger and Spacy's pos tagger. The next set of models were utilized for their word embeddings, both were from word2vec; however, one was trained on the Brown Corpus, the other on Google News. The most robust models were the sentiment analysis models. The first was a neural network with an embedding layer and LSTM layer while the Bayes model was Multinomial Naive

3.1 Continuous Bag of Words

The purpose of our Word2Vec Continuous Bag of Words model is to create word embeddings so that we can map the painting labels to be consistent with the emotion labels present in our sentiment analysis classifiers. An example of this purpose would be that the first painting is labeled 'humility' and 'optimism'. Since neither of these emotions are present in either dataset that we trained the sentiment analysis classifiers with, we will use word embeddings to produce the most similar emotion and thus give this painting a new emotion that is present in our classifiers. This allows us to take the user's input, determine their input's sentiment, then return a painting with similar emotions.

3.2 Preprocessing

A tokenizer was created along with encoded X value sentences for training, testing, and validation. We used the NLTK part of speech tagger and Spacy tagger for tagging nouns in painting titles, so we could add them to the list of objects in each painting. To account for words that have not been seen by the object detector, we used the Brown corpus embeddings and Google Word2Vec embeddings to map unknown objects to known objects seen by the object detector. The GO emotion labels do not neatly line up with the painting emotion labels; our original solution was to use word embeddings to obtain the most similar emotion. This approach yielded the undesirable effect of certain words being mapped to completely different emotions due to these words being close to each other vectorically. The solution was to manually map embeddings ourselves, and move the embeddings to mapping objects. We found that similar nouns are more likely to be near each other vectorically than similar emotions.

3.3 Naive Bayes

For our Naive Bayes model, we used the scikit-learn library's Multinomial Naive Bayes model for training and prediction. Initially, we saw that the OneVsRest classifier model was used for classification with fitting single classes against all other classes. However, rather than having a single column be the column of target values, we had 29 categories (emotions) that were all going to be classified. After deciding that 29 classifications would be an inefficient use of our time, we found that we could instead consolidate all 29 categories into a single column which mapped the original emotion label associated with the text to a label ID. For example, admiration is the leftmost category in the dataset, so it is mapped to label ID 1. To prepare our data, we first imported the Go Emotions dataset, which contains a large collection of various texts, each classified with a single emotion label, with 29 emotions in total. However, in the original dataset, some texts lacked a label altogether, while many were labeled with 'neutral'. As neither of these cases are useful to us, we removed the rows which did not contain any emotion classification. Next, in the interest of time and testing, we set a range of 200 rows as training data. Had we had more time, it would have been more optimal to train on more data to improve performance. Then, with sklearn's feature_extraction library, we used CountVectorizer() for text preprocessing and tokenizing, which links the index value of a word in the vocabulary to its frequency in the whole training corpus. While occurrence count was a good starting point, the issue was that longer documents would have higher average count values than shorter documents, so to avoid these potential discrepancies, it sufficed to divide the number of occurrences of each word in a document by the total number of words in the document. This brings us to tf-idf, or Term Frequency times Inverse Document Frequency, also under sk-learn feature_extraction. Using this, we fit the TfidfTransformer to our estimator to the data, and then transform our count-matrix to a tf-idf representation. Once we had our features, we trained a sk-learn naive_bayes classifier, MultinomialNB, to try to predict the outcome on a new text input. To do this, we needed to extract the features using a nearly identical feature extraction chain as before, except that we called transform() rather than fit_transform() on the transformers, as they had already been fitted to the training set.

3.4 Neural Network

The neural network, used for sentiment analysis, was constructed using the Keras sequential architecture. The key focus points of this model include the embeddings layer, LSTM layer, and GO dataset. The initial task of the first model was to predict a single sentiment from text, multiclass.. In this early model we utilized the emotions dataset that was made up of only 'I' statements. The performance of this model after being trained multiple times adjusting activation layers, embeddings, learning rates, always was abysmal. This is most likely due to the dataset's poor quality and not being a good fit for the task of classifying non first person sentences. After continuous failure we decided to throw the model out due to its failure and limited scope in regards to only producing a single label, while paintings had multiple emotions. The new goal of the new model was to predict a range of sentiments given text input. This task is formally known as a multilabel task. With a total of 27 possible emotions, it was not expected for this model to perform exceptionally well, due to the complexity of mapping all these labels at once. However, utilizing an LSTM layer with 500 word embeddings, we were able to achieve reasonable success, roughly 40% top categorical accuracy of k = 3. An LSTM layer (Long Short-Term Memory) is a recurrent neural network that utilizes sequence order to better learn and classify tasks. Prior to implementation of this layer, the embedding layer paired with a relu activation layer was peaking at roughly 10% accuracy. Beginning with a LSTM layer of 128, we began testing all the way up to 512. Anything above 512 took far too long to train while anything below didn't reach the accuracy threshold we were aiming for. Our conclusion as to why an LSTM layer bolstered performance by so much was due to how well it performs on sequences, which is essentially what this classification relies on. In regards to further adjustment and tinkering with layers, we found that adding any other layers besides embeddings and an LSTM layer hurt the performance. We additionally increased word embeddings from 200 to 500 gradually to edge out the performance. Overall, despite the difficulty of the task this neural network delivers moderate success and works well when being used to classify inputs for our generation. The major drawback of this model is that usually after its top 3 labels, it begins to perform exceptionally badly, going as far as having the 4th prediction having nothing to do with the prior 3. Perhaps reducing the amount of labels and stacking models would be a more appropriate approach with better accuracy, at the cost of being more computationally expensive.

4. OBJECT DETECTION

We took every painting and ran them through an object detection algorithm. The goal of this was to add another set of labels for each painting. This would allow us to query paintings that had similar objects to what the user input so that we could retrieve paintings that made more sense. While some paintings did accurately capture the emotion the users typed in, we wanted the objects to also be present. To get past the simplicity of our object detection, we used word embeddings to map unseen objects to objects that had been seen.

5. PERFORMANCE

5.1 Production

Overall, utilizing all the models results in acceptable performance. The main flow of the program is first a user enters input and the tagger identifies nouns. Next, the embeddings model gets similar nouns that have been seen by the object detector. This allows a larger query of objects to be returned. Next, after all the paintings with at least a single matching object are returned, we run the input through our sentiment analysis model. Finally, the paintings with the most matching objects, and the best matching sentiment is returned to the user.

5.2 Results

Our criteria for acceptable performance is being able to identify paintings with related objects while getting sentiments to somewhat match. The best combination of parameters for our model are the Neural network as the sentiment classifier, Google news as the word embeddings model, and Spacy as the POS tagger. The most significant variation can be seen between the different sentiment classifiers followed by the different embeddings. Due to the brown embedding model being far less developed and having a far smaller vocab list than the google embeddings, it is expected that this is the inferior embedding model. As for the comparison of neural networks to naive bayes, the amount of time spent training in addition to the amount of data utilized in training was a clear indication of what the performance would be like.

5.3 Exploration

Something to note; however, is that a naive Bayes classifier spreads out across every label, one for is the sentence angry, one for is the sentence sad, etc… could lead to the most promising results albeit more inefficient. This idea is something we would have liked to test. Besides the removal of the first emotions dataset, another robust change in approach includes the decision to note us word embeddings to map sentiments between data sets. While some inputs produce reasonable results, for example this painting was brought forth by the prompt: “Give me the fire emblem or you will die a swift death by my sword”. There are multiple person objects recognized with relatively strong confidence, as well as the presence of weapons and death.

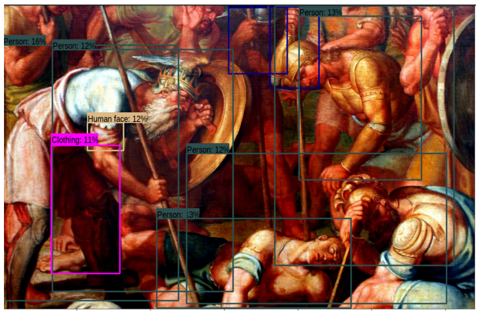

Figure 1. Object detected image #1

Other inputs struggle to find a good painting to match. Here, the given input was: “The cat ran up the hill into the hut after it almost got hit by a car”

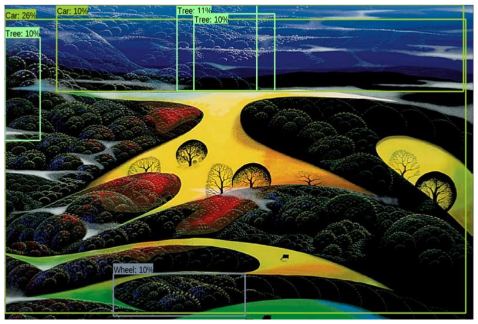

Figure 2. Object detected image #2

Here the sentiment was approval, excitement, and joy- which is a bit off, and the object detection completely failed. The main limitations are our small pool of paintings, our lack of paintings with a diverse set of nouns, our lack of variety of paintings with similar nouns, our object detection, and our dataset of paintings.

6. FUTURE DIRECTIONS

Had time permitted, there would have been additional functionality we intended to implement. However, we left it out of the initial scope of this project, as it was not as closely related to the NLP topics and techniques we had covered throughout the semester, and thus was not our main focus. With an available software known as “Deep Painterly Harmonization”, we would include an option for users to generate stock images in addition to the originally generated painting, and merge them with image harmonization. This would allow objects unseen by the object detector to appear in paintings. Another way we could expand on this project is to add more models on top of our Naive Bayes and Neural Network models. We could then compare more models to find the best performance. This could be easily implemented by adding a trained model to the file, implementing the prediction, and adding it as an option to use for prediction.

7. REFERENCES

[1] Go Emotions Dataset https://www.kaggle.com/datasets/shivamb/go-emotions-google-emotions-dataset

[2] Emotions Dataset https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp

[3] https://code.google.com/archive/p/word2vec/

[4] https://keras.io/api/layers/recurrent_layers/lstm/

[5] https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/

[6] Multinomial Naive Bayes documentation Working With Text Data — scikit-learn 1.0.2 documentation

[7] PAINTINGS DATASET: http://saifmohammad.com/WebPages/wikiartemotions.html