Emotion Recognition in Tweets

CPSC 571

Lotus Pearl Borromeo
Jiro Go
Md Mushfiqur Rahman

Group 16

Introduction

@Intro

Our approach tries to achieve emotion recognition in tweets using preprocessing techniques and multinomial naive bayes, with potential applications in social media analytics and customer satisfaction analysis.

Introduction

...

@InitialPlan

The initial plan was to use a MET Gala dataset and use database optimizations along with the Naive Bayes algorithm to facilitate emotion recognition in tweets.

During the project proposal presentation the following feedback were given:

- Specify which emotion model will be used.
- Review database optimization literature and specify which techniques will be used during the development.

Introduction

@ThePlan

The second plan was to use a pre-existing dataset that already includes tweets and the emotion it conveys. Furthermore, figure out a novel way to label tweets since the database optimization route is no longer an option. The group accomplished the following:

- Found a pre-existing dataset containing tweets and its emotion label.
  - The found dataset includes: the tweet ID, the emotion found in the tweet, and the tweet content.
  - "Emotion Detection from Text" from Kaggle (39 827 tweets)
- Developed an algorithm that uses Naive Bayes to label tweets.
  - We realized during the development of this that improving the preprocessing algorithm could potentially increase the accuracy of our algorithm.

The Initial Algorithm

@Overview

The overview of the algorithm used to achieve emotion recognition in tweets is as follows:

- Preprocessing of tweets
  - Removal of special characters, stop words, and converting to lowercase
- Splitting the dataset
  - Training Set (80%)
  - Testing Set (20%)
- Convert the tweet content into a bag-of-words representation
- Used MultinomialNB classifier on the training set
- Predict the emotion labels of the tweets in the testing set
  - Anger, Boredom, Empty, Enthusiasm, Fun, Happiness, Hate, Love, Neutral, Relief, Sadness, Surprise, Worry
- Evaluate the performance of the classifier
  - sklearn.metrics (precision, recall, f1-scores, accuracy)

Preprocessing Techniques

@RemovalOfSpecialCharactersAndPunctuation

Removing special characters and punctuation from the text can help reduce noise and improve the performance of the model.

Preprocessing Techniques

@StopWordsRemoval

Stop words are common words that do not carry much meaning, such as "the," "and," and "in." Removing stop words from the text can reduce the dimensionality of the data and improve the performance of the model.

Classification Report 1.0

@BaseCode

```
Number of training samples: 32000
Number of testing samples: 8001
Accuracy: 0.31121109861267343
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        20
     boredom       0.00      0.00      0.00        40
       empty       0.00      0.00      0.00       151
  enthusiasm       0.00      0.00      0.00       161
         fun       0.00      0.00      0.00       344
   happiness       0.33      0.31      0.32      1034
        hate       0.25      0.01      0.01       260
        love       0.51      0.31      0.38       766
     neutral       0.32      0.37      0.34      1655
      relief       0.00      0.00      0.00       354
     sadness       0.27      0.10      0.15      1046
    surprise       0.00      0.00      0.00       440
       worry       0.29      0.70      0.41      1730

    accuracy                           0.31      8001
   macro avg       0.15      0.14      0.12      8001
weighted avg       0.26      0.31      0.26      8001

Number of correctly classified samples: 2490
```

Accuracy Improvement Attempts

@FutureChanges

- Modifying Preprocessing techniques
  - Stemming
  - Handling Negation
  - Removing Hashtags and Mentions
- Boosting
- Using a different model (CNN) instead of naive bayes
- Modifying size of the dataset

Modifying Preprocessing Techniques

@Stemming

These techniques involve reducing words to their base or root form. For example, "running" and "ran" can both be stemmed to "run." This can help reduce the number of unique features in the data and improve the performance of the model.

Classification Report 2.0

@Stemming

```
Number of training samples: 32000
Number of testing samples: 8001
Accuracy: 0.3125859267591551
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        20
     boredom       0.00      0.00      0.00        40
       empty       0.00      0.00      0.00       151
  enthusiasm       0.00      0.00      0.00       161
         fun       0.00      0.00      0.00       344
   happiness       0.33      0.29      0.31      1034
        hate       0.17      0.00      0.01       260
        love       0.53      0.30      0.38       766
     neutral       0.31      0.39      0.35      1655
      relief       0.00      0.00      0.00       354
     sadness       0.30      0.11      0.16      1046
    surprise       1.00      0.00      0.00       440
       worry       0.29      0.70      0.41      1730

    accuracy                           0.31      8001
   macro avg       0.23      0.14      0.12      8001
weighted avg       0.32      0.31      0.26      8001

Number of correctly classified samples: 2501
```

Modifying Preprocessing Techniques

@HandlingNegation

Negation can invert the polarity of words in a sentence, e.g., "not happy" is different from "happy." Techniques such as adding a negation tag before the affected words can help the model distinguish between the two.

Classification Report 3.0

@HandlingNegation

```
Number of training samples: 32000
Number of testing samples: 8001
Accuracy: 0.31508561429821275
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        20
     boredom       0.00      0.00      0.00        40
       empty       0.00      0.00      0.00       151
  enthusiasm       0.00      0.00      0.00       161
         fun       0.00      0.00      0.00       344
   happiness       0.33      0.29      0.31      1034
        hate       0.29      0.01      0.01       260
        love       0.52      0.30      0.38       766
     neutral       0.31      0.39      0.35      1655
      relief       0.00      0.00      0.00       354
     sadness       0.32      0.11      0.17      1046
    surprise       1.00      0.00      0.00       440
       worry       0.29      0.70      0.41      1730

    accuracy                           0.32      8001
   macro avg       0.24      0.14      0.13      8001
weighted avg       0.33      0.32      0.26      8001

Number of correctly classified samples: 2521
```

Modifying Preprocessing Techniques

@RemovalOfHashtags&Mentions

Tweets often contain **#hashtags** and **@mentions**, which can provide important context for emotion detection. These can be extracted and treated as separate features in the model.


CCF Hematology Oncology Fellows @ccfhemonc · Apr 10

Congratulations to all authors on their recent publication highlighting the downstream effects of #COVID19 on US #hematology #oncology trainees, including impacts on professional development, clinical access, mentorship, and job searches. @HemOncFellows

Classification Report 4.0

@RemovalOfHashtags&Mentions

```
Number of training samples: 32000
Number of testing samples: 8001
Accuracy: 0.31508561429821275
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        20
     boredom       0.00      0.00      0.00        40
       empty       0.00      0.00      0.00       151
  enthusiasm       0.00      0.00      0.00       161
         fun       0.00      0.00      0.00       344
   happiness       0.33      0.29      0.31      1034
        hate       0.29      0.01      0.01       260
        love       0.52      0.30      0.38       766
     neutral       0.31      0.39      0.35      1655
      relief       0.00      0.00      0.00       354
     sadness       0.32      0.11      0.17      1046
    surprise       1.00      0.00      0.00       440
       worry       0.29      0.70      0.41      1730

    accuracy                           0.32      8001
   macro avg       0.24      0.14      0.13      8001
weighted avg       0.33      0.32      0.26      8001

Number of correctly classified samples: 2521
```

Boosting

@OptimizationTechnique

- AdaBoost algorithm can be seen as an optimization technique that improve the accuracy of Naive Bayes by iteratively training and weighting multiple instances of the Naive Bayes classifier.

- In each iteration, the AdaBoost algorithm selects the instances that were misclassified in the previous iteration and gives them a higher weight in the training process. This allows the Naive Bayes classifier to focus on the more difficult samples and improve its accuracy over time.

Boosting

@Caveats

- However, it is important to note that AdaBoost can sometimes be sensitive to noise and outliers in the data, which can negatively impact the performance of the model.

- Requires a very large dataset

- The choice of weak learner can affect the performance of AdaBoost, and other algorithms may be more suitable depending on the specific task and data

Classification Report 5.0

@ADABoostClassifier

```
Number of training samples: 32000
Number of testing samples: 8001
Accuracy: 0.2597175353080865
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        20
     boredom       0.00      0.00      0.00        40
       empty       0.00      0.00      0.00       151
  enthusiasm       0.00      0.00      0.00       161
         fun       0.00      0.00      0.00       344
   happiness       0.41      0.01      0.01      1034
        hate       0.00      0.00      0.00       260
        love       0.63      0.09      0.15       766
     neutral       0.24      0.83      0.37      1655
      relief       0.00      0.00      0.00       354
     sadness       0.67      0.00      0.00      1046
    surprise       0.00      0.00      0.00       440
       worry       0.30      0.36      0.33      1730

    accuracy                           0.26      8001
   macro avg       0.17      0.10      0.07      8001
weighted avg       0.32      0.26      0.16      8001

Number of correctly classified samples: 2078
```

Exploring A Different Model

@CNN

In the context of optimization, we wanted to explore alternative models to Naive Bayes for the task of classifying emotions in tweets. To this end, we explored Convolutional Neural Network (CNN) model as a potential solution.

- CNN (Convolutional Neural Network) is a deep learning algorithm that is often used for image and signal processing tasks.
- It works by applying filters to the input data, producing a set of feature maps that capture different aspects of the input.
- CNN's ability to automatically learn and capture complex patterns and relationships in the input data makes it well-suited for a wide range of image and signal processing tasks

Exploring A Different Model

@CNNvsNBComparision

CNNs are often more effective than Naive Bayes for image or signal processing tasks because they can capture complex patterns and relationships in the data. However, Naive Bayes can be more effective in natural language processing and also in cases where the features are independent of each other or when there is limited data available.

Exploring A Different Model

@CNNBenefits

The potential benefits of using CNNs for emotion recognition include their ability to capture complex patterns and relationships in the data.

- more robust to noise and variations in the input data,
  - making them better suited for recognizing emotions in different contexts and across different individuals.
- CNNs can be trained to automatically learn relevant features from the input
  - reduces the need for manual feature engineering
  - make the approach more scalable.

Exploring A Different Model

@CNNDrawbacks

One potential drawback of using CNNs for emotion recognition is

- they may require a larger amount of labeled training data to achieve good performance compared to Naive Bayes.

- CNNs are often computationally more expensive and require more resources to train and deploy than Naive Bayes.

CNN Algorithm Overview

@CNN

- Tokenization and Padding
- Splitting into Training and Testing Sets
- Building the CNN model
- Compiling the model
- Training the model
- Evaluating the Performance

```
# Tokenize the text and pad the sequences
max_words = 10000
tokenizer = Tokenizer(num_words=max_words, lower=True)
tokenizer.fit_on_texts(df['content'])

X = tokenizer.texts_to_sequences(df['content'])
X = pad_sequences(X, padding='post')
y = pd.get_dummies(df['sentiment']).values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the CNN model
embedding_dim = 100
model = Sequential([
    Embedding(max_words, embedding_dim, input_length=X.shape[1]),
    Conv1D(128, 5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(30, activation='relu'),
    Dense(y.shape[1], activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

# Train the CNN model
epochs = 5
batch_size = 32
model.fit(X_train, y_train, epochs=epochs, batch_size=batch_size, validation_split=0.1)

# Evaluate the performance
y_pred = model.predict(X_test)
y_pred_mapped = np.argmax(y_pred, axis=1)
y_test_mapped = np.argmax(y_test, axis=1)
```

Classification Report 6.0

@CNNClassifier

```
Accuracy: 0.2815898012748406
Classification report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        20
           1       0.00      0.00      0.00        40
           2       0.02      0.01      0.01       151
           3       0.00      0.00      0.00       161
           4       0.09      0.06      0.07       344
           5       0.26      0.29      0.28      1034
           6       0.20      0.16      0.18       260
           7       0.41      0.32      0.36       766
           8       0.33      0.40      0.36      1655
           9       0.14      0.06      0.08       354
          10       0.25      0.37      0.30      1046
          12       0.07      0.05      0.06       440
          13       0.32      0.32      0.32      1730

    accuracy                           0.28      8001
   macro avg       0.16      0.16      0.16      8001
weighted avg       0.26      0.28      0.27      8001
```

Decision! Decision!

@stats

Learning model
Naive Bayes
Accuracy
32%

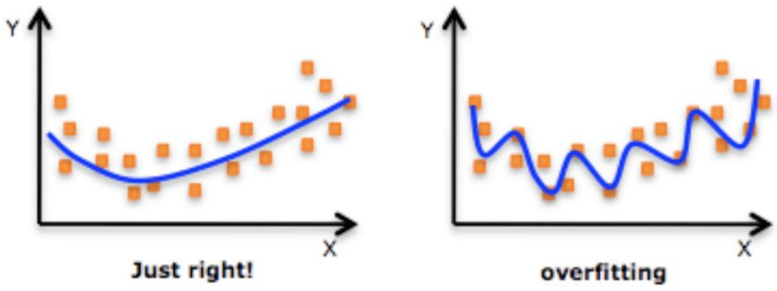
Learning model
Convolutional Neural Network (CNN)
Accuracy
28%

After applying Naive Bayes and CNN models to the same dataset, we made a decision to move forward with Naive Bayes based solely on the metric of accuracy.

Increasing Dataset Size

@Dataset

- Reduced overfitting
- Improved estimation of probabilities
- Improved coverage of feature space
- Reduced bias



A dataset ("cleaned Emotion Extraction dataset from twitter" from Kaggle) containing around 848,000 unique tweet values that were initially classified into 3 emotion labels was preprocessed in order to be added to the existing dataset used.

Preprocessing the dataset included :

1. Changing the emotion labels to be the same as the original dataset
   - Disappointed → Sadness, Happy → Happiness, Angry → Anger
2. Combining the 2 CSV files of the datasets into one CSV file [1][2]

Classification Report 7.0

@IncreasingDatasetSize

```
Number of training samples: 478979
Number of testing samples: 119745
Accuracy: 0.8027391540356591
Classification report:
              precision    recall  f1-score   support

       anger       0.00      0.00      0.00        25
     boredom       0.00      0.00      0.00        33
       empty       0.00      0.00      0.00       144
  enthusiasm       0.00      0.00      0.00       158
         fun       0.00      0.00      0.00       396
   happiness       0.83      0.90      0.86     37883
        hate       0.85      0.76      0.80     36975
        love       0.48      0.01      0.03       750
     neutral       0.34      0.02      0.03      1683
      relief       0.00      0.00      0.00       310
     sadness       0.75      0.87      0.80     39224
   sentiment       0.00      0.00      0.00         1
    surprise       0.00      0.00      0.00       419
       worry       0.23      0.01      0.02      1744

    accuracy                           0.80    119745
   macro avg       0.25      0.18      0.18    119745
weighted avg       0.78      0.80      0.78    119745
```

Code Snippet Gallery

@gallery

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(df['content'], df['sentiment'], test_size=0.2, random_state=42)

# Print the number of training and testing samples
print('Number of training samples:', len(X_train))
print('Number of testing samples:', len(X_test))

# Convert the tweet content into a bag-of-words representation
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)

# Train a Naive Bayes classifier on the training set
classifier = MultinomialNB()
classifier.fit(X_train, y_train)

# Predict the sentiment labels of the tweets in the testing set
y_pred_mapped = classifier.predict(X_test)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred_mapped)
print('Accuracy:', accuracy)
print('Classification report:\n', classification_report(y_test, y_pred_mapped))

# Create a DataFrame with the testing data and predicted emotion
# Add the predicted sentiment to the original DataFrame
df_test = pd.DataFrame({'content': df.iloc[y_test.index]['content'].values, 'predicted_sentiment': y_pred_mapped})
df_test['id'] = df.iloc[y_test.index]['id'].values
df_test['sentiment'] = df.iloc[y_test.index]['sentiment'].values

# Save the predicted sentiment DataFrame to a CSV file
df_test.to_csv('comparisonNaive.csv', index=False, columns=['id', 'sentiment', 'content', 'predicted_sentiment'])

# Print the number of correctly classified samples
num_correct = (y_test == y_pred_mapped).sum()
print('Number of correctly classified samples:', num_correct)
```

References

@references

[1]“Home.” Kaggle, PASHUPATI GUPTA, https://www.kaggle.com/datasets/pashupatigupta/emotion-detection-from-text?resource=download. Accessed 18 March 2023.

[2]“Home.” Kaggle, https://www.kaggle.com/datasets/kosweet/cleaned-emotion-extraction-dataset-from-twitter. Accessed 1 April 2023.

[3] https://emotions.clevertap.com/ , a platform utilizing AI and machine learning to analyze and predict consumer emotions

[4] https://www.tagtog.com/ , which is an online platform that uses artificial intelligence and machine learning algorithms to perform text annotation and information extraction tasks

[5] https://www.lighttag.io/ , which is a collaborative platform that uses AI-powered tools to perform text annotation and data labeling tasks

Thank you!

Questions?