Text Classification-Who is author of Hamlet

Submitted by Manish Bafna
Student Id: 19655
Instructor: Dr. Henry Chang
GIT: **Text Classification**

Table of Content

Introduction

Shakespeare Conspiracy Theories: Shakespeare is one of the most well-known and studied playwrights in history, but despite this, there have been a number of conspiracy theories surrounding his life and work. Some of the most popular conspiracy theories include:

Shakespeare did not write his own plays and sonnets: This theory argues that Shakespeare was not educated or cultured enough to have written the works attributed to him, and that they were actually written by someone else, such as Francis Bacon, Christopher Marlowe, or Edward de Vere, the Earl of Oxford.

Shakespeare was a front for a secret group of writers: This theory suggests that Shakespeare was part of a larger group of writers who collaborated on the plays and sonnets, but that he was chosen to be the public face of the group.

While there is no conclusive evidence to support any of these conspiracy theories, they continue to fascinate and intrigue many people.

Introduction

Text Classifier to predict the real author of Hamlet:

In recent years, there has been interest in using machine learning techniques to analyze Shakespeare's works and determine whether he was the sole author or whether he collaborated with others. One approach is to use a text classifier to predict the real author of a particular work, such as Hamlet.

A text classifier is a type of machine learning model that can be trained to identify patterns in text and classify it into different categories. In the case of Shakespeare's works, a text classifier can be trained on a set of texts written by different authors, and then used to predict the authorship of a new text based on the patterns it finds.

Introduction

To train a text classifier for authorship attribution, researchers typically use a variety of features, such as the frequency of certain words or phrases, the use of particular grammatical constructions, and the overall style and tone of the text. Once the classifier is trained, it can be used to make predictions about the authorship of a new text.

While text classifiers are not foolproof, and there is always some degree of uncertainty in their predictions, they can provide a useful tool for analyzing Shakespeare's works and shedding light on the question of authorship.

Design

Step 1: Data Collection and Preprocessing

Collect a dataset of texts written by potential authors of Hamlet, including Christopher Marlowe, William Stanley, and Francis Bacon. Preprocess the data by cleaning and normalizing the text, removing stop words, and converting it into a format suitable for machine learning.

Step 2: Feature Extraction

Extract features from the preprocessed text data that will be used to train the text classifier.

Step 3: Model Selection and Training

Choose a machine learning model for the text classifier, such as a Naive Bayes classifier

Split the preprocessed dataset into training and validation sets, and train the model using the training data. Evaluate the performance of the model using the validation data, and tune its hyperparameters as necessary.

Design

Step 4: Testing and Prediction

Once the model is trained, use it to predict the authorship of Hamlet by inputting its features into the trained model.

The output of the model will be the predicted author of Hamlet, based on the patterns and features extracted from the training data.

Step 5: Evaluation and Iteration

Evaluate the performance of the model on the test data, and iterate on the feature extraction, model selection, and hyperparameter tuning as necessary to improve the accuracy of the text classifier.

Implement

Manually calculation

Data set

	Doc	Words	Author
Training	1	W1 W2 W3 W4 W5	C (Christopher Marlowe)
	2	W1 W1 W4 W3	C (Christopher Marlowe)
	3	W1 W2 W5	C (Christopher Marlowe)
	4	W5 W6 W1 W2 W3	W (William Stanley)
	5	W4 W5 W6	W (William Stanley)
	6	W4 W6 W3	F (Francis Bacon)
	7	W2 W2 W4 W3 W5 W5	F (Francis Bacon)
Test	8 (Hamlet)	W1 W4 W6 W5 W3	?

Implement

Manually calculation

Priors:

P(X) = The probability of an Author X

= Number of Author X / total number of Authors = NX / N

Note:

P(C) = The probability of Author C = 3/7 (i.e., 3 C- Authors / total Authors)

P(W) = The probability of Author W = 2/7 (i.e., 2 W- Authors / total Authors)

P(F) = The probability of Author F = 2/7 (i.e., 2 F- Authors / total Authors)

Implement

Manually calculation

Conditional probabilities:

P(w|x) = If a document belongs to Author x, the probability that the document has word w.

= The probability the word w appears on the Author x document.

= (count(w, x) + 1) / (count(x)+|V|)

Implement

Manually calculation

Conditional probabilities:

Original definition of P(w|x) = count(w, x) / count (x)

count(w, x): how many times the word w appears on the x Author documents.

count(x): how many words on the x Author documents.

|V|: number of vocabularies = number of different words

Tunable knobs (i.e., parameters) of Naive Bayes

1 and |V| are used for Laplace Smoothing to prevent the possibility of letting P(w|x) have value of 0 or 1. Other values can be used to replace 1 and |V|.

Implement

(V in this case = 6)

The Test only has 5 words: W1, W4 , W6, W5, W3.

- P(W1|C) = The probability the word "W1" appear on the Author “C” documents.

= (count(W1, C) + 1) / (count(C)+|V|)

= (4+1) / (12+6) = 5/18

- P(W1|W) = The probability the word "W1" appear on the Author “W” documents.

= (count(W1, W) + 1) / (count(W)+|V|)

= (1+1) / (8+6) = 2/14

Implement

P(W1|F) = The probability the word "W1" appear on the Author “F” documents.

= (count(w1, F) + 1) / (count(F)+|V|)

= (0+1)/(9+6) = 1/15

Similiarly,

P(W3|C) = (count(W3, C) + 1) / (count(C)+|V|) = (2+1)/(12+6) = 3/18

P(W3|W) = (count(W3, W) + 1) / (count(W)+|V|) = (1+1)/(8+6) = 2/14

P(W3|F) = (count(W3, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15

P(W4|C) = (count(W4, C) + 1) / (count(C)+|V|) = (2+1) / (12+6) = 3/18

Implement

P(W4|W) = (count(W4, W) + 1) / (count(W)+|V|) = (1+1)/(8+6) = 2/14

P(W4|F) = (count(W4, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15

P(W5|C) = (count(W5, C) + 1) / (count(C)+|V|) = (2+1) / (12+6) = 3/18

P(W5|W) = (count(W5, W) + 1) / (count(W)+|V|) = (2+1)/(8+6) = 3/14

P(W5|F) = (count(W5, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15

P(W6|C) = (count(W6, C) + 1) / (count(C)+|V|) = (0+1)/(12+6) = 1/18

P(W6|W) = (count(W6, W) + 1) / (count(W)+|V|) = (2+1)/(8+6) = 3/14

P(W6|F) = (count(W6, F) + 1) / (count(F)+|V|) = (1+1)/(9+6) = 2/15

Test (Manual)

Decide whether d8 (i.e., document 8) belongs to Author C or Author W or Author F

- Step 1: Analysis

A. The probability of d8 (i.e., document 8) belonging to Author C

$P(C|d8) = P(C) * P(d8|C) / P(d8)$

==> Applying Bayes Theorem

$= P(C) * P(W1 \cap W4 \cap W6 \cap W5 \cap W3 \mid C) / P(d8)$

==> Applying Naive Bayes Theorem

$\propto (P(C) * P(W1|C) * P(W4|C) * P(W6|C) * P(W5|C) * P(W3|C)) / P(d8)$

$= P(C) * P(W1|C) * P(W4|C) * P(W6|C) * P(W5|C) * P(W3|C) / P(d8)$

Test (Manual)

==> Applying Compare Model

$P(C|d8) \propto P(C) * P(W1|C) * P(W4|C) * P(W6|C) * P(W5|C) * P(W3|C)$

$= 3/7 * 5/18 * 3/18 * 1/18 * 3/18 * 3/18 = 0.00003061924$

B. The probability of document 8 belonging to Author W.

==> Applying Naive Bayes Theorem

$P(W|d8) \propto (P(W) * P(W1|W) * P(W4|W) * P(W6|W) * P(W5|W) * P(W3|W)) / P(d8)$

$= P(W) * P(W1|W) * P(W4|W) * P(W6|W) * P(W5|W) * P(W3|W) / P(d8)$

==> Applying Compare Model

$P(W|d8) \propto P(W) * P(W1|W) * P(W4|W) * P(W6|W) * P(W5|W) * P(W3|W)$

$= 2/7 * 2/14 * 2/14 * 3/14 * 3/14 * 2/14 = 0.00003824936$

Test (Manual)

C. The probability of document 8 belonging to Author F.

==> Applying Naive Bayes Theorem

$P(F|d8) \propto (P(F) * P(W1|F) * P(W4|F) * P(W6|F) * P(W5|F) * P(W3|F)) / P(d8)$

$= P(F) * P(W1|F) * P(W4|F) * P(W6|F) * P(W5|F) * P(W3|F) / P(d8)$

==> Applying Compare Model

$P(F|d8) \propto P(F) * P(W1|F) * P(W4|F) * P(W6|F) * P(W5|F) * P(W3|F)$

$= 2/7 * 1/15 * 3/15 * 2/15 * 3/15 * 3/15 = 0.00002031746$

Implement Colab

```
[2] from google.colab import files
    uploaded = files.upload()
```

Choose Files Text_Classifier.csv

- **Text_Classifier.csv**(text/csv) - 142 bytes, last modified: 3/4/2023 - 100% done

```
Saving Text_Classifier.csv to Text_Classifier.csv
```

```
data = pd.read_csv("Text_Classifier.csv")
print(data)
```

```
   Doc              Words Author
0    1     W1 W2 W3 W4 W5       C
1    2        W1 W1 W4 W3       C
2    3           W1 W2 W5       C
3    4     W5 W6 W1 W2 W3       W
4    5           W4 W5 W6       W
5    6           W4 W6 W3       F
6    7  W2 W2 W4 W3 W5 W5       F
```

Implement Colab

Load English tokenizer, tagger, parser, NER and word vectors

```
[212] parser = English()
```

Define a function to tokenize text using the spacy parser and remove stopwords and punctuation

```
[216] def spacy_tokenizer(text):
          # Parse the text using the spacy parser
          tokens = parser(text)

          # Remove stopwords and punctuation from the tokens
          tokens = [token.lemma_.lower().strip() if token.lemma_ != "-PRON-" else token.lower_ for token in tokens if token.text not in stop_words and token.text not in string.punctuation]

          # Return the cleaned tokens
          return tokens
```

Define a transformer class for cleaning text

```
[225] class TextPreprocessor(TransformerMixin):
          def transform(self, X, **transform_params):
              # Apply the clean_text function to each element of X
              return [clean_text(text) for text in X]
```

Implement Colab

Define a function to clean the text

```
[226] def clean_text(text):
          return text.strip().lower()
```

Create a Bag of Words vectorizer using the spacy tokenizer

```
[227] bow_vectorizer = CountVectorizer(tokenizer=spacy_tokenizer, ngram_range=(1,1))
```

Create a TF-IDF vectorizer using the spacy tokenizer

```
[228] tfidf_vectorizer = TfidfVectorizer(tokenizer=spacy_tokenizer)
```

Split the data into training and testing sets

```
[246] X_train, X_test, y_train, y_test = train_test_split(data['Words'], data['Author'], test_size=0.5)
```

Implement Colab

+ Code + Text

Logistic Regression Classifier

```
[247] classifier = LogisticRegression()
```

Define a pipeline that preprocesses text, vectorizes it using the bag-of-words approach, and classifies it using logistic regression

```
[248] pipe = Pipeline([
          ('preprocessor', TextPreprocessor()),
          ('vectorizer', bow_vectorizer),
          ('classifier', classifier)
      ])
```

Fit the pipeline to the training data

```
[250] pipe.fit(X_train, y_train)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn(
```

Pipeline

TextPreprocessor

CountVectorizer

Test Colab

Predict the authors of the test data using the trained model

```
[251] predicted_authors = pipe.predict(X_test)
```

Predict the author of new data using the trained model

```
[252] new_text = ["w1 w4 w6 w5 w3"]
```

```
[253] predicted_author = pipe.predict(new_text)
      print("Predicted author:", predicted_author)
```

```
Predicted author: ['W']
```

Enhancement Ideas

Use a more advanced machine learning algorithm: The provided example uses a simple approach of calculating probabilities for each word and author. More advanced machine learning algorithms, such as deep learning neural networks, can be implemented to improve the accuracy of the text classifier.

Use more training data: The current dataset is small and only contains a few words per document. A larger and more diverse training dataset would help the text classifier learn more effectively and improve its accuracy.

Use pre-trained language models: Pre-trained language models such as BERT, GPT, or XLNet have been shown to achieve state-of-the-art performance on many natural language processing tasks. Using a pre-trained language model as a starting point could help improve the accuracy of the text classifier.

Conclusion

Document 8(Hamlet) should belong to the Author W ((William Stanley))

The same has been calculated manually - Author W

The same has been achieved successfully through the code - Author W

References

Shakespeare: the conspiracy theories (telegraph.co.uk)

Shakespeare authorship question - Wikipedia

Text classification and Naive Bayes (stanford.edu)

Text Classifier (sfbu.edu)