NLP class predictor

11 March 2023 01:26

The task is to classify news articles into different categories, such as politics, entertainment, sports, business, world news, and tech. The dataset used for this task is the **News Category Dataset** from the Kaggle. This dataset contains around 210k news headlines from 2012 to 2022 from HuffPost.

Content

Each record in the dataset consists of the following attributes:

- category: category in which the article was published.
- headline: the headline of the news article.
- authors: list of authors who contributed to the article.
- link: link to the original news article.
- short_description: Abstract of the news article.
- date: publication date of the article.

Preprocessing steps: The dataset was preprocessed in the following steps:

1. Removal of punctuations and special characters
2. Conversion of all text to lowercase
3. Tokenization of the text using the NLTK library
4. Removal of stopwords using the NLTK library
5. Stemming of the words using the Porter stemmer algorithm

```
# Load the stopwords and initialize the stemmer and lemmatizer
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
def preprocess_text(text):
    # Lowercase the text
    text = text.lower()
    # Remove punctuations
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords and stem or lemmatize the words
    words = [stemmer.stem(word) for word in tokens if word not in stop_words]
    # words = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    # Join the words back into a string
    processed_text = ' '.join(words)
```

```
    # words = [lemmatizer.lemmatize(word) for word in tokens if word not in stop_words]
    # Join the words back into a string
    processed_text = ' '.join(words)
    return processed_text
```

Architecture of the model:

A convolutional neural network (CNN) was employed for this challenge. An embedding layer, two convolutional layers, two max-pooling layers, and two dense layers made up the CNN. The categorical cross-entropy loss function and the Adam optimizer were used to train the model. By adjusting the learning rate and batch size, the model was refined.

Evaluation metrics and results:

The model was evaluated using the following metrics: accuracy, precision, recall, and F1-score. The model achieved an accuracy of 88%, with the following values for precision, recall, and F1-score for each category

	Prcesion	Recall	F1-Score	Support
Politics	0.93	0.92	0.92	7095
Entertainment	0.90	0.92	0.91	3537
Sports	0.83	0.89	0.85	386
Busniess	0.75	0.68	0.72	1209
World News	0.71	0.79	0.75	642
Tech	0.75	0.65	0.70	419
Accuracy			0.88	13888
Macro Avg.	0.81	0.81	0.81	13888
Weighted Avg.	0.88	0.88	0.88	13888

Discussion of performance and possible improvements:

The model did a decent job of categorising news articles into several groups. There is room for improvement, though. Using a larger dataset can help the model learn more features and generalise more effectively, which can enhance performance. The model can also be

improved further by adjusting the hyperparameters, such as the number of filters, kernel sizes, and layers.

Sample predictions and explanations:

Exapmple 1

```
article = "Apple unveils new iPhone"
predicted_category = predict_category(article)
print(predicted_category)
```

```
TECH
```

Example 2

```
article = "Donald Trump announces presidential campaign"
predicted_category = predict_category(article)
print(predicted_category)
```

```
POLITICS
```

Example 3

```
article = "Serena Williams wins Wimbledon"
predicted_category = predict_category(article)
print(predicted_category)
```

```
SPORTS
```