HyperionDev

Capstone Project – NLP Applications

Task

Visit our website

Introduction

Great job getting to this capstone project! In this project, you will be applying natural language processing (NLP) to sentiment analysis.

Developer portfolio

Developers who have the edge are those who find ways to apply their newfound skills from the get-go. As you may know, a **developer portfolio**, which is a collection of online creations that you have made, allows you to demonstrate your skills rather than just telling people about them. It's a way of bringing your CV to life and introducing yourself to the world. As you learn more skills and put these into practice, each project that you complete will become more efficient and eye-catching.

These capstone projects give you the means to create projects for your very own developer portfolio, allowing you to walk away from this course not only with a certificate but, more importantly, with a head start to your tech career!

NLP applications

NLP applications can be classified into a number of categories based on what they do. Let's have a brief look at some of these categories.

Language translation

The volume of information being shared online is growing at an incredible rate, but due to language barriers, not everything is accessible to everyone. Language translation helps us conquer these language barriers by facilitating the translation of technical manuals, support content, or websites quickly and inexpensively. The challenge with language translation technologies is not in translating words, but in understanding the meaning of sentences to provide a true translation. If you've ever watched a movie in one language with subtitles in another language where you have been able to understand both languages, you will realise how prone to error the process of translation can be, even when using human experts. Sometimes the subtitles are outright wrong, and sometimes, although the words are translated accurately, the nuance of the meaning is lost or miscommunicated. Really effective NLP language translation is a challenging area in which exciting progress is being made.

Text classification

One of the applications of NLP that we experience on a daily basis is the text classification in our email folders. By using predefined categories, we can organise our spam folders and inboxes so that we can access relevant emails or messages more efficiently.

Automatic summarisation

When working with huge amounts of information (like articles, books, and websites), it can be extremely useful to be able to shorten these pieces into condensed forms that only show the pieces of information that are most useful to you. This is what automatic summarisation is about. According to Expert.ai (2020): “Automatic summarization is relevant not only for summarizing the meaning of documents and information, but also for understanding the emotional meanings inside the information, such as in collecting data from social media.”

Sentiment analysis

Similar to how we can infer someone’s meaning from their tone, sentiment analysis allows us to detect the emotion behind a piece of text using NLP. This is particularly useful for large companies who want to know what the general sentiment is that people hold towards their company. By analysing articles and write-ups about their company using sentiment analysis, people can gain a fairly accurate idea of how others feel about the company based on the language used when the company is being discussed.

Question answering

This is an application of NLP that has come a long way in a short space of time. Simply put, these systems allow a computer to answer a question posed by a human. Siri and Google Assistant are well-known voice-based question-answering systems, but text-based systems can now be seen on almost every banking or online shopping site in the form of a chatbot. ChatGPT is a relatively advanced implementation of a question-answering model.

The task at hand

In this task, you will develop a Python program that performs sentiment analysis on a dataset of product reviews.

Instructions

A key focus of this project will be ensuring that your code is correct, well-formatted, readable, and that it adheres to the **PEP 8 style guide**. In this regard, **make sure that you do the following** (and double-check before submitting your work to avoid losing marks unnecessarily):

1. Identify and remove all syntax, runtime, and logical errors from your code.
2. Construct readable code. To ensure this, add comments to your code, use descriptive variable names, and make good use of whitespace and indentation.
3. Create modular code by creating functions to perform specific units of work.
4. Write efficient code. How you choose to write code to create the solution to the specified problem is up to you. However, make sure that you write your code as efficiently as possible.
5. Make sure that all outputs that your program provides to a user are easy to read and understand. Label all data that you output.

Practical task

1. Create a code file named **sentiment_analysis**. You may use a **.py** or **.ipynb** file.
2. Load the following Amazon product reviews dataset collated by Datafiniti from **Kaggle**:

   Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv

3. Develop a Python script for sentiment analysis. Remember to include informative comments that clarify the rationale behind each line of code. Within the script, you will perform the following tasks using the spaCy library:

3.1. **Implement a sentiment analysis model using spaCy:** Load the `en_core_web_md` spaCy model to enable NLP tasks. This model will help you analyse and classify the sentiment of the product reviews.

3.2. **Preprocess the text data:** Remove stop words, and perform any necessary text cleaning to prepare the reviews for analysis.

3.2.1. To select the “reviews.text” column from the dataset and retrieve its data, you can simply use the square brackets notation. Here is the basic syntax:

```
reviews_data = dataframe['reviews.text']
```

This column represents the feature variable containing the product reviews we will use for the sentiment analysis.

3.2.2. To remove all missing values from this column, you can simply use the **dropna()** function from pandas using the following code:

```
clean_data = dataframe.dropna(subset=['reviews.text'])
```

3.3. **Create a function for sentiment analysis:** Define a function that takes a product review as input and predicts its sentiment.

3.4. **Test your model on sample product reviews:** Test the sentiment analysis function on a few sample product reviews to verify its accuracy in predicting sentiment.

4. **Write a brief report or summary** that includes the following:

4.1. A description of the dataset used.
4.2. Details of the preprocessing steps.
4.3. Evaluation of results.
4.4. Insights into the model's strengths and limitations.

Export it as a PDF file named **sentiment_analysis_report.pdf**.

Additional instructions:

- Some helpful guidelines on cleaning text:
    - To remove stop words, you can utilise the `.is_stop` attribute in spaCy. This attribute helps identify whether a word in a text qualifies as a stop word or not. Stop words are common words that do not add much meaning to a sentence, such as “the”, “is”, or “of”. Subsequently, you can employ the filtered list of tokens or words (words with no stop words) to conduct sentiment analysis.

    - You can also make use of the `lower()`, `strip()`, and `str()` methods to perform some basic text cleaning.

- You can use the spaCy model and the `.sentiment` attribute to analyse the review and determine whether it expresses a positive, negative, or neutral sentiment. To use the `.polarity` attribute, you will need to install the **TextBlob library**. To install the TextBlob library and download additional data, run the following commands from the terminal:

```
# Install spacytextblob
pip install spacytextblob

# Download additional data for TextBlob
python -m textblob.download_corpora
```

- Once you have installed TextBlob, you can use the `.sentiment` and `.polarity` attribute to analyse the review and determine whether it expresses a positive, negative, or neutral sentiment. When you are ready to determine sentiment, add the code below to your **sentiment_analysis** script:

```
# Using the polarity attribute
polarity = doc._.blob.polarity

# Using the sentiment attribute
sentiment = doc._.blob.sentiment
```

  FYI: The underscore in the code just above is a **Python convention** for naming private attributes. Private attributes are not meant to be accessed directly by the user, but can be accessed through public methods.

- You can use the `.polarity` attribute to measure the strength of the sentiment in a product review. A polarity score of 1 indicates a very positive sentiment, while a polarity score of -1 indicates a very negative sentiment. A polarity score of 0 indicates a neutral sentiment.

- You can also use the `similarity()` function to compare the similarity of two product reviews. A similarity score of 1 indicates that the two reviews are more similar, while a similarity score of 0 indicates that the two reviews are not similar.
    - For example, you could choose two product reviews from the "reviews.text" column and compare their similarity. To select a specific review from this column, simply use indexing, as shown in the code below:

```
my_review_of_choice = data['reviews.text'][0]
```

    - The above code retrieves a review from the “reviews.text” column at index 0. Ensure you do not use an index that is out of bounds, meaning it exceeds the number of data points or rows in our dataset.

Important: Be sure to upload all files required for the task submission inside your task folder and then click “Request review” on your dashboard.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.

Reference list

Expert.ai Team. (2020, June 9). Natural language processing applications. *Expert.ai*. **https://expertsystem.com/natural-language-processing-applications/**