Intro2NLP_Assignment_2_Your_Name

November 10, 2024

1 Introduction to Natural Language Processing: Assignment 2

In this exercise we'll practice features extraction using Tf-Idf and SpaCy as well as multiclass text classification using the word embedding technique.

- You can use built-in Python packages, spaCy, scikit-learn, Numpy and Pandas.
- Please comment your code
- Submissions are due Tuesdays at 23:59 **only** on eCampus: **Assignmnets » Student Submissions » Assignment 3 (Deadline: 12.11.2024, at 23:59)**
- Name the file aproppriately: "Assignment_2_<Your_Name>.ipynb" and submit only the Jupyter Notebook file.
- If you are working in a group of two, please have the names of both of the members in the file name.
- Please use relative path, your code should work on my computer if the Jupyter Notebook and the file are both in the same directory.

Example: file_name = bbc-news.csv, **DON'T use:** /Users/ComputerName/Username/Documents/.../bbc-news.csv

1.0.1 Task 1 (2 points)

Write a function `extract_proper_nouns(my_file_name)` that takes a file name (my_file_name.txt) as input and returns a list containing all proper nouns with more than one token.

Example:

text = "Honk Kong and Japan are two countries in Asia and New York is the largest city in the world"

return = `["New York", "Hong Kong"]` **(Note: it should not return "Japan")**

```
[ ]: def extract_proper_nouns(my_file_name):
         several_token_propn = []
         # here comes your code
         return(several_token_propn)
```

1.0.2 Task 2 (3 points)

Write a function `common_lemma(my_file_name)` that takes a file name (my_file_name.txt) as input and returns a Python dictionary with lemmas as `key` and the `value` that should contain a list with both verbs and nouns sharing the same lemma.

Examples:

1. text = "When users google for a word or any query, their system internally runs a pipeline in order to process what the person is querying."

return = `{"query": ["query", "querying"]}`

2. text = I really loved the movie and show, the movie was showing reality but it showed sometimes nonesense!

return = `{"show": ["show", "showing", "showed"]}` **(Note: it should not return "movie" because both "movie"s are NOUN)**

[ ]:
```
def common_lemma(my_file_name):
    tokens_with_common_lemma = {}
    # here comes your code
    return(tokens_with_common_lemma)
```

1.0.3 Task 3 (1 point)

Load the data `bbc-text.csv`; This datata consists of 2225 documents from the BBC news website corresponding to stories from 2004-2005.

[ ]:
```
# Here comes your code
```

1.0.4 Task 4 (1 point)

Show how many articles we have for each topical area (class label) in the dataset using a plot.

[ ]:
```
# Here comes your code
```

1.0.5 Task 5 (2 point)

Preprocessing: Define two following functions and apply them to the dataset: 1. Remove punctuation 2. Remove any numbers

[ ]:
```
def remove_punctuation(corpus):
    # Here comes your code
    return(cleaned_corpus)

def remove_numbers(corpus):
    # Here comes your code
    return(cleaned_corpus)
```

1.0.6 Task 6.1 (1 points)

Split the data into training and test set (70% and 30%) using scikit-learn, shuffle it, and set the `seed=101 (random_state)`.

NOTE: If working with this dataset is not computationaly possible for you, you can work only with a subset of the dataset (i.e., the first 1000 rows) and use only the first 150 tokens for each article. You should point this out in your code.

[ ]:
```
# Here comes your code
```

1.0.7 Task 6.2 (3 points)

1.0.8 Training models on TF-IDF vectors:

a) Convert each article in your data splits to a vector representation using the tf-idf-vectorizer.

b) Using the vectors from the previous step, train the `MLPClassifier` and another model of your choice from the scikit-learn library.

c) Test both of your models on the test set from Task 6.1.

[ ]:
```
# Here comes your code
X_train_tfidf = ...

y_pred_MLP_tfidf = ...
y_pred_<MODEL>_tfidf = ...
```

1.0.9 Task 6.3 (3 points)

1.0.10 Training models on SpaCy model vector representation:

a) Convert each article in your data splits to a vector representation using the pre-trained spaCy model. (**Hint:** It should be stored as an array)

b) Using the vectors from the previous step, train the `MLPClassifier` and another model of your choice from the scikit-learn library.

c) Test both of your models on the test set from Task 6.1.

[ ]:
```
# Here comes your code
X_train_spacy = ...

y_pred_MLP_spacy = ...
y_pred_<MODEL>_spacy = ...
```

1.0.11 Task 7 (4 points)

Using the predictions from the four classifiers, evaluate the models and report accuracy, recall, precision, f1 scores and confusion matrix for each of them. (**Hint:** You should build a confusion matrix for multi-class classification)

[ ]:
```
# Here comes your code
```