Elio Desnoulez (19129131)

NLP Assignment1 Review

Assignement Goals

- Learn how to predict tags for posts from StackOverflow.
- Use multilabel classification approach.

Libraries Used

- Numpy — a package for scientific computing.
- Pandas — a library providing high-performance, easy-to-use data structures and data analysis tools for the Python
- scikit-learn — a tool for data mining and data analysis.
- NLTK — a platform to work with natural language.

Text preprocessing

- First step : TextPrepare
  - We needed to lowercase the text
  - Replace all these characters by spaces :
    ( ) { } [ ] | @ , ;
  - Remove these characters :
    ^ 0-9 a-z ' ' # + _
  - Remove all the english stopwords downloaded from nltk

Text preprocessing

- **Results :**
  - SQL Server - any equivalent of Excel's CHOOSE function? -> sql server equivalent excels choose function
  -
  - How to free c++ memory vector<int> * arr? ->
  - free c++ memory vectorint arr
- "Basic tests are passed." is printed as return of test_text_prepare() function.

```
def text_prepare(text):
    """
        text: a string
        return: modified initial string
    """
    text = text.lower() # lowercase text
    text = re.sub(REPLACE_BY_SPACE_RE, ' ', text) # replace REPLACE_BY_SPACE_RE symbols by space in text
    text = re.sub(BAD_SYMBOLS_RE, '', text) # delete symbols which are in BAD_SYMBOLS_RE from text

    # delete stopwords from text
    filtered_text = ""
    tokenized_text = word_tokenize(text)
    for i in range(len(tokenized_text)):
        if not tokenized_text[i] in STOPWORDS:
            filtered_text += tokenized_text[i];
            if i != len(tokenized_text) - 1:
                filtered_text += ' '

    return filtered_text
```

Text preprocessing

- Second step : **WordsTagsCount**
  - We needed to count all words and tags to sort them after and find the most used words / tags.
  - Create a Bag of words representation

Count Words and Tags

- **Results :**
  - Most common Tags are javascript, c#, java
  - Most common Words are using, c, php

```
# Dictionary of all words from train corpus with their counts.
print("Counting words and tags :")
words_counts = {}
tags_counts = {}

words=[]
tags=[]
for i in range(0, len(X_train)):
    if i % 1000 == 0 :
        print(((i / 1000) + 1), "%")
    words = words + (re.findall(r'\w+', X_train[i])) # cantain all the words in the dataset
    tags = tags + y_train[i] # contain tags present in the dataset
print("Finished")

words_counts = Counter(words) # Create word map of occurences
tags_counts=Counter(tags) # Create tags map of occurences

most_common_tags = sorted(tags_counts.items(), key=lambda x: x[1], reverse=True)[:3]
most_common_words = sorted(words_counts.items(), key=lambda x: x[1], reverse=True)[:3]
```

Transforming text to a vector

- **Results :**
  - words_to_index = {'hi': 0, 'you': 1, 'me': 2, 'are': 3}
  - examples = ['hi how are you']
  - answers = [[1, 1, 0, 1]]
- "Basic tests are passed." is printed as return of test_my_bag_of_words() function.

```
DICT_SIZE = 5000
WORDS_TO_INDEX = {}
INDEX_TO_WORDS = {}

most_common_words = sorted(words_counts.items(), key=lambda x: x[1], reverse=True)[:5000]
for i in range(0, DICT_SIZE):
    WORDS_TO_INDEX[most_common_words[i][0]] = i
    INDEX_TO_WORDS[i] = most_common_words[i][0]


def my_bag_of_words(text, words_to_index, dict_size):
    """
        text: a string
        dict_size: size of the dictionary

        return a vector which is a bag-of-words representation of 'text'
    """
    result_vector = np.zeros(dict_size)
    words = text.split(" ")
    for i in range(0, len(words)):
        for key, value in words_to_index.items():
            if words[i] == key :
                result_vector[words_to_index[key]] += 1
    return result_vector
```

Task 3 : BagOfWords

```
row = X_train_mybag[10].toarray()[0]

non_zero_elements_count = 0
for i in range(0, DICT_SIZE):
    if (row[i] == 1):
        non_zero_elements_count += 1

print('BagOfWords : ', non_zero_elements_count)
```

```
X_train shape  (100000, 5000)
X_val shape  (30000, 5000)
X_test shape  (20000, 5000)
BagOfWords :  6
```

Encountered problems

- I don't know well Python yet (I code usually in Bash, Web, JavaScript, C, C++, Java).
- Not much time to do the assignement as I got Distributed Computing courses/project in the same time.
- Running the program takes so much time :
  - i7-4600U CPU @ 2.10GHz
    - 1227.74s user 58.21s system 99% cpu 21:35.73 total -> 22 minutes in local
  - Xeon(R) CPU E3-1245 V2 @ 3.40GHz
    - Real 16m56.822s user 16m56.022s sys 0m1.056s -> 17 minutes on server
- Maybe with Jupyter I could win some time if I understood it correctly.