HW 2

Due Feb 6 by 11:59pm **Points** 10 **Submitting** a text entry box or a file upload **Available** until Feb 6 at 11:59pm

This assignment was locked Feb 6 at 11:59pm.

Part a: Bag of Words

Transforming text to a vector

Machine Learning algorithms work with numeric data and we cannot use the provided text data "as is". There are many ways to transform text data to numeric vectors. In this task you will try to use two of them.

Bag of words

One of the well-known approaches is a *bag-of-words* representation. To create this transformation, follow the steps:

1. Find *N* most popular words in train corpus and numerate them. Now we have a dictionary of the most popular words.
2. For each title in the corpora create a zero vector with the dimension equals to *N*.
3. For each text in the corpora iterate over words which are in the dictionary and increase by 1 the corresponding coordinate.

Let's try to do it for a toy example. Imagine that we have *N* = 4 and the list of the most popular words is

```
['hi', 'you', 'me', 'are']
```

Then we need to numerate them, for example, like this:

```
{'hi': 0, 'you': 1, 'me': 2, 'are': 3}
```

And we have the text, which we want to transform to the vector:

```
'hi how are you'
```

For this text we create a corresponding zero vector

```
[0, 0, 0, 0]
```

And iterate over all words, and if the word is in the dictionary, we increase the value of the corresponding position in the vector:

```
'hi':  [1, 0, 0, 0]
'how': [1, 0, 0, 0] # word 'how' is not in our dictionary
'are': [1, 0, 0, 1]
'you': [1, 1, 0, 1]
```

The resulting vector will be

```
[1, 1, 0, 1]
```

Implement the described encoding in the function *my_bag_of_words* with the size of the dictionary equals to 4.

```
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)
    ######################################
    ######### YOUR CODE HERE #############
    ######################################
    return result_vector
```

Part b: TF-IDF

1. Test the script tfidf_demo.ipynb in the Jupiter note and make sure they work.

2. Replace the movie review data "texts" in the script file with your own defined document and test it.

3. Given the below documents.

```
texts = [
    "good movie", "not a good movie", "did not like",
    "i like it", "good one"
]
```

Given the definition of TF and IDF, what is the **sum** of TF-IDF values for 1-grams in "good movie" text? Enter a math expression as an answer.