ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НИЖЕГОРОДСКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Р. Е. АЛЕКСЕЕВА

*Кафедра «Прикладная математика»*

Лабораторная работа №3

по дисциплине «Базы данных»

Тема: «Работа с текстом: категории, мешок слов, tf-idf, ЕЯ(NLP)»

**Студент**

_____ Валькова.Н.П.
(Подпись) (Фамилия, И., О.)

18-ПМ ……
(Группа) (Дата сдачи)

**Проверил**

_____ Моисеев А.Е
(Подпись) (Фамилия, И., О.)

Отчет защищен «___» _____2021_г.
с оценкой_____

Нижний Новгород, 2021

## Оглавление

1.Введение.....3
2.Постановка задачи.....4
3. Решение.....5

**Введение:**

**Мешок слов** (*Bag of Words*) - это модель текстов на натуральном языке, в которой каждый документ или текст выглядит как неупорядоченный набор слов без сведений о связях между ними. Его можно представить в виде матрицы, каждая строка в которой соответствует отдельному документу или тексту, а каждый столбец — определенному слову. Ячейка на пересечении строки и столбца содержит количество вхождений слова в соответствующий документ. Название «мешок» происходит из-за игнорирования порядка токенов в рассматриваемом документе. Так, два документа, отличающиеся лишь порядком токенов, будут иметь одинаковые векторы.

**TF-IDF** — статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции. Мера TF-IDF часто используется в задачах анализа текстов и информационного поиска , например, как один из критериев релевантности документа поисковому запросу, при расчёте меры близости документов при кластеризации.

**Обработка естественного языка** (*Natural Language Processing, NLP*) — пересечение машинного обучения и математической лингвистики, направленное на изучение методов анализа и синтеза естественного языка. Сегодня NLP применяетсяво многих сферах, в том числе в голосовых помощниках, автоматических переводах текста и фильтрации текста. Основными тремя направлениями являются: распознование речи(*Speech Recognition*), понимание естественного языка (*Natural Language Unders*) и генерация естественного языка (*Natural Language Generation*).

**Постановка задачи:**
Сделать мешок слов для текста при помощи sklearn.feature_extraction.text.CountVectorizer.
Посчитать метрики TF-IDF для документов с отзывами IMDB при помощи sklearn.feature_extraction.text.TfidfTransformer.
Вывести наиболее употребляемы слова из датасетов.

**Решение**

- Скачиваем базы данных  Reviews (отзывы на корм для животных) и spam с сайта kaggle.com

Импортируем:
```
import numpy as np
import pandas as pd


from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
```

- Читаем данные из файла 'Reviews.csv' и  'spam.csv' заполняем наши массивы данными из этого файла.

```
data = pd.read_csv('Reviews.csv' , nrows = 1000)
data1 = pd.read_csv('spam.csv' , nrows = 1000)
```

- Используя CountVectorizer из модуля sklearn, находим мешок слов для исходных текстов отзывов Так же с помощью TfidfTransformer из модуля sklearn, строим TF-IDF для исходных данных.

- Создаем мешок слов, считаем метрики TF-IDF

```
bag = count.fit_transform(data.Text)
bag_tfidf = tfidf.fit_transform(data.Text)
```

- Ищем топ 10 слов для отзывов на корм

```
df = tfidf.fit_transform(data['Text'])
ind = np.argsort(tfidf.idf_)[::-1]
feat = tfidf.get_feature_names()
top_n = 10
top_features = [feat[i] for i in ind[:top_n]]
print(top_features)
```

**Результат работы:**

Для 1000 строк из reviews

```
..    ...  ...                                                 ...
995   996  ...  BLACK MARKET HOT SAUCE IS WONDERFUL.... My hus...
996   997  ...  Man what can i say, this salsa is the bomb!! i...
997   998  ...  this sauce is so good with just about anything...
998   999  ...  Not hot at all. Like the other low star review...
999  1000  ...  I have to admit, I was a sucker for the large ...

[1000 rows x 7 columns]
Word bag =  [[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]


tf-idf [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
['îţ', 'horseradish', 'humans', 'hue', 'hu', 'html', 'hr', 'howdy', 'housewarming', 'hotter']
```

Результат для всех данных reviews

```
[568454 rows x 7 columns]
['îţ', 'perplexion', 'ardous', 'perphaps', 'perpetuity', 'freshmixers', 'freshmorels', 'areaas', 'b002we2lou', 'freshnes']
```

Топ слов для 1000 строк из spam

```
Топ слов для спама, отмеченного как хамство

['zaher', 'hit', 'helen', 'hell', 'hella', 'hence', 'hep', 'heron', 'hes', 'hesitate']
Топ слов для спама

['zouk', 'dave', 'goals', 'give', 'girls', 'getzed', 'getting', 'getstop', 'germany', 'genuine']
```

Топ слов для всех данных spam

```
Топ слов для спама, отмеченного как хамство

['zyada', 'images', 'imat', 'imf', 'imin', 'immed', 'immunisation', 'impede', 'implications', 'importantly']
Топ слов для спама

['zouk', 'dirtiest', 'disaster', 'divorce', 'dizzee', 'dob', 'doggin', 'dogs', 'doit', 'donate']
```

**Листинг:**

```
import numpy as np
import pandas as pd
from nltk.tokenize import RegexpTokenizer
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

data = pd.read_csv('Reviews.csv')
data.drop(data.columns[[2, 3, 4]], axis = 1, inplace = True)
print(data)
count = CountVectorizer()
tfidf = TfidfVectorizer()

bag = count.fit_transform(data.Text)
bag_tfidf = tfidf.fit_transform(data.Text)

# print("Word bag = ",bag.toarray())
# print('\n')
# np.set_printoptions(precision = 2)
# # print(repr(count.vocabulary))
# print('tf-idf',bag_tfidf.toarray())

df = tfidf.fit_transform(data['Text'])
ind = np.argsort(tfidf.idf_)[::-1]
feat = tfidf.get_feature_names()
top_n = 10
top_features = [feat[i] for i in ind[:top_n]]
print(top_features)

data1 = pd.read_csv('spam.csv' )
data1.drop(data1.columns[[2, 3, 4]], axis = 1, inplace = True)
Ham = data1[data1.v1 == 'ham']
Spam = data1[data1.v1 == 'spam']

count = CountVectorizer()
tfidf = TfidfVectorizer()

# bag = count.fit_transform(Ham.v2)
# bag_tfidf = tfidf.fit_transform(Ham.v2)
#
# bag = count.fit_transform(Spam.v2)
# bag_tfidf = tfidf.fit_transform(Spam.v2)
# print("Word bag = ",bag.toarray())
# print('\n')
# np.set_printoptions(precision = 3)
# print('tf-idf',bag_tfidf.toarray())

df = tfidf.fit_transform(Ham['v2'])
ind = np.argsort(tfidf.idf_)[::-1]
feat = tfidf.get_feature_names()
top_n = 10
top_features = [feat[i] for i in ind[:top_n]]
print("Топ слов для спама, отмеченного как хамство\n")
```

```
print(top_features)

df_1 = tfidf.fit_transform(Spam['v2'])
ind = np.argsort(tfidf.idf_)[::-1]
feat = tfidf.get_feature_names()
top_n = 10
top_features = [feat[i] for i in ind[:top_n]]
print("Топ слов для спама\n")
print(top_features)
```