Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Аналіз даних в інформаційно-управляючих системах

ЗВІТ

до лабораторної роботи №8

Виконав
студент **ІП-01 Хернуф Валід Алі-Еддін**
(№ групи, прізвище, ім’я, по батькові)

Київ 2022

Опис задачі

1) провести очистку текстових даних від стоп-слів/тегів/розмітки;
2) виконати токенізацію текстових елементів;
3) провести лематизацію текстових елементів;
4) порахувати метрику TF-IDF для 10 слів, що найчастіше зустрічаються в корпусі;
5) створити та наповнити Bag of Word для всіх нормалізованих слів.

Опис вирішення задачі та екрані форми результатів

Після імпортування усіх потрібних бібліотек, та зчитування набору даних з текстами (рис. 1), ми приступили до очистки та токенізацію текстових даних.

```
def read_dataset(path, separ, en):
    return p.read_csv(path, sep = separ, encoding = en)

dFrame = read_dataset("ukr_text.csv", ",", "utf-8")
```
✓ 0.1s

Рис. 1 – Зчитування набору даних

Спочатку ми поділи на токени, одразу перевіряю текст за шаблоном, який залишає символи апострофів та тире. (рис. 3)

```
def to_tokens(dFrame):
    for i in range(dFrame.shape[0]):
        for j in range(dFrame.shape[1]):
            temp_text = dFrame.iloc[i, j]
            dFrame.iloc[i, j] = re.sub("[^А-Яа-яЁёЇїІіЄєҐґ'\-\s]", "", temp_text).split()

to_tokens(dFrame)
dFrame.info()
dFrame.head(5)
```
✓ 0.4s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1122 entries, 0 to 1121
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Title   1122 non-null   object
 1   Body    1122 non-null   object
dtypes: object(2)
memory usage: 17.7+ KB
```

	Title	Body
0	[Кличко, покликав, німецьких, інвесторів, до, ...	[Київ, -, перспективний, і, відкритий, ринок, ...
1	[З'явилося, відео, як, байкер, почав, стріляти...	[З'явилося, відео, конфлікту, між, мотоцикліст...
2	[У, центрі, Києва, посеред, вулиці, помер, чол...	[У, Києві, на, Бессарабській, площі, вранці, в...
3	[Нічний, ураган, перетворив, Хрещатик, на, смі...	[Київ, вночі, серпня, пережив, найсильнішу, гр...
4	[Потоп, у, Києві, столицю, накрив, ураган, з, ...	[Уночі, Київ, вкотре, накрила, негода, Найсиль...

Рис. 2 – Токенізація та перевірка за шаблоном (з результатами)

Далі ми імпортували список всіх стоп-слів та додали до нього ще деякі потрібні слова. (рис. 3)

```python
def csv_to_list(path, n_l, enc):
    with open(path, newline = n_l, encoding = enc) as s_w:
        return [element for sublist in list(csv.reader(s_w)) for element in sublist]

def get_stop_words(path, n_l, enc):
    stop_words = csv_to_list(path, n_l, enc)
    stop_words.remove("stopword")
    stop_words.extend(["тис", "грн", "вул", "сек", "хв", "обл", "кв", "пл", "напр", "гл", "о", "зам", "із"])
    return stop_words

stop_words = get_stop_words("stop_words_ua.csv", "", "utf-8")
print(stop_words)
```

✓ 0.4s

Рис. 3 – Імпортування та додавання стоп-слів

Наступним кроком було виправлення помилок, які могли виникнути про токенізації через залишення деяких символів. Для їх вирішення ми використовували певний шаблон. Після цього ми перевели усі слова до нижнього регістру та видалили слова, що співпадали зі стоп-словами. (рис. 4)

```python
def fix_porblems(dFrame, stop_words):
    for i in range(dFrame.shape[0]):
        for j in range(dFrame.shape[1]):
            for k in range(len(dFrame.iloc[i,j])):
                dFrame.iloc[i,j][k] = dFrame.iloc[i,j][k].lower()
            dFrame.iloc[i,j] = [x for x in dFrame.iloc[i,j] if (re.fullmatch("(\w+(\-|\'|\`)\w+|\w+)", x) and len(x) > 2)]
            dFrame.iloc[i,j] = [x for x in dFrame.iloc[i,j] if not (x in stop_words)]

fix_porblems(dFrame, stop_words)
dFrame.info()
dFrame.head(5)
```

✓ 16.3s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1122 entries, 0 to 1121
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Title   1122 non-null   object
 1   Body    1122 non-null   object
dtypes: object(2)
memory usage: 17.7+ KB
```

	Title	Body
0	[кличко, покликав, німецьких, інвесторів, києва]	[київ, перспективний, відкритий, ринок, бізнес...
1	[з'явилося, відео, байкер, почав, стріляти, во...	[з'явилося, відео, конфлікту, мотоциклістом, в...
2	[центрі, києва, вулиці, помер, чоловік]	[києві, бессарабській, площі, вранці, четвер, ...
3	[нічний, ураган, перетворив, хрещатик, смітник]	[київ, вночі, серпня, пережив, найсильнішу, гр...
4	[потоп, києві, столицю, накрив, ураган, градом]	[уночі, київ, вкотре, накрила, негода, найсиль...

Рис. 4 – Виправлення помилок та видалення стоп-слів

Після усіх попередніх кроків ми можемо провести лематизацію текстових елементів. (рис. 5)

```python
def lematize(dFrame):
    uk_l = pymorphy2.MorphAnalyzer(lang='uk')
    for i in range(dFrame.shape[0]):
        for j in range(dFrame.shape[1]):
            for k in range(len(dFrame.iloc[i,j])):
                dFrame.iloc[i,j][k] = uk_l.parse(dFrame.iloc[i,j][k])[0].normal_form

lematize(dFrame)
dFrame.info()
dFrame.head(5)
```

✓ 33.6s

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1122 entries, 0 to 1121
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Title   1122 non-null   object
 1   Body    1122 non-null   object
dtypes: object(2)
memory usage: 17.7+ KB
```

	Title	Body
0	[кличко, покликати, німецький, інвестор, київ]	[кий, перспективний, відкритий, ринок, бізнес,...
1	[з'явитися, відео, байкер, почати, стріляти, в...	[з'явитися, відео, конфлікт, мотоцикліст, воді...
2	[центр, київ, вулиця, померти, чоловік]	[кий, бессарабський, площа, вранці, четвер, се...
3	[нічний, ураган, перетворити, хрещатик, смітник]	[кий, вночі, серпень, пережити, найсильніший, ...
4	[потоп, кий, столиця, накрити, ураган, град]	[уночі, кий, вкотре, накрити, негода, найсильн...

Рис. 5 – Лематизація текстових елементів

Далі нам потрібно порахувати метрики TF-IDF для десяти слів, що найчастіше зустрічаються. Вочевидь буде те, що першим кроком буде знаходження саме цих слів. (рис. 6)

```python
def get_top_words(dFrame):
    bodies = []
    for i in range(dFrame.shape[0]):
        bodies += dFrame.iloc[i]["Body"]
    return dict(collections.Counter(bodies).most_common(10))

top_words_dict = get_top_words(dFrame)
print(top_words_dict)
top_words = list(top_words_dict.keys())
```

✓ 0.1s Python

```
{'україна': 1195, 'новий': 589, 'компанія': 574, 'корреспондент': 571, 'країна': 455, 'світ': 449, 'український': 444, 'повідомлятися': 433, 'перший': 426, 'один': 394}
```

Рис. 6 – Знаходження 10 слів, що найчастіше зустрічаються

Наступним кроком буде підрахунок метрики TF-IDF для знайдених слів. (рис. 7-8)

```python
def identity_tokenizer(text):
    return text

def df_tfidf(dFrame, topWords):
    tokenized_list = []
    for i in range(dFrame.shape[0]):
        tokenized_list.append(dFrame.iloc[i]["Body"])
    tfidf = TfidfVectorizer(tokenizer = identity_tokenizer, lowercase = False)
    vectors = tfidf.fit_transform(tokenized_list)
    feature_names = tfidf.get_feature_names_out()
    denselist = vectors.todense().tolist()
    d_l, f_n = top_n_tfidf(feature_names, denselist, topWords)
    return p.DataFrame(n.array(d_l).T.tolist(), columns = f_n)

def top_n_tfidf(feature_names, denselist, topWords):
    f_n = []
    d_l = []
    feature_names = n.array(feature_names)
    denselist = n.array(denselist)
    for i in range(len(feature_names)):
        is_there = False
        for e2 in topWords:
            if feature_names[i] == e2:
                is_there = True
        if is_there:
            f_n.append(feature_names[i])
            temp = []
            for j in range(len(denselist[:,i])):
                temp.append(denselist[j][i])
            d_l.append(temp)
    return d_l, f_n

df = df_tfidf(dFrame, top_words)
df
```

✓ 3.4s

Рис. 7 – Знаходження метрик TF-IDF

	компанія	корреспондент	країна	новий	один	перший	повідомлятися	світ	україна	український
0	0.074764	0.000000	0.039572	0.017326	0.054679	0.018513	0.015480	0.000000	0.059960	0.075369
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.068100	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.065389	0.000000	0.055535	0.000000	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.036248	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...	...
1117	0.093566	0.032873	0.015238	0.013344	0.021056	0.007129	0.000000	0.021802	0.040406	0.050790
1118	0.013783	0.036723	0.014591	0.000000	0.013441	0.034131	0.000000	0.000000	0.071853	0.006948
1119	0.053115	0.040432	0.033736	0.039389	0.041436	0.000000	0.000000	0.000000	0.017039	0.010709
1120	0.043648	0.022151	0.184820	0.000000	0.028376	0.028822	0.000000	0.088146	0.151691	0.029334
1121	0.008649	0.039505	0.054937	0.040090	0.042173	0.085673	0.000000	0.008734	0.027747	0.087195

1122 rows × 10 columns

Рис. 8 – Метрики TF-IDF

Останнім кроком виконання цієї лабораторної роботи було створення та наповнення Bag of Word. Саме це ми й зробили. (рис. 9-10)

```python
def calculateBOW(wordset, l_doc):
    tf_diz = dict.fromkeys(wordset, 0)
    for word in l_doc:
        tf_diz[word] = l_doc.count(word)
    return tf_diz

def get_wordset(dFrame):
    wordset = None
    for i in range(dFrame.shape[0]):
        if (i == 1):
            wordset = n.union1d(dFrame.iloc[0]["Body"], dFrame.iloc[1]["Body"])
        elif (i > 1):
            wordset = n.union1d(wordset, dFrame.iloc[i]["Body"])
    return wordset

def dataFrame_bowl(dFrame):
    bow = []
    wordset = get_wordset(dFrame)
    for i in range(dFrame.shape[0]):
        bow.append(calculateBOW(wordset, dFrame.iloc[i]["Body"]))
    return p.DataFrame(bow)

df_bow = dataFrame_bowl(dFrame)
df_bow.head(10)
```

✓ 1m 22.8s

Рис. 9 – Створення Bag of Word

	а-ля	аахен	аахенський	абаджян	абатство	абдель	абдул-джаббар	абдулла	абдумалик	абер	...	ґас	ґедун	ґодар	ґрунт	ґрунтовий	ґрунтовний	ґрунтовн
0	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	

Рис. 10 – Виведення 10 перших рядків Bag of Word

Через те, що виведення цього Bag of Word не э дужу інформативним, якщо починати з початку алфавіту, я вирішив також вивести його для слів, що найчастіше зустричаются. (рис. 11)

```
df_useful = df_bow[top_words]
df_useful.head(15)
```

✓ 0.7s

	україна	новий	компанія	корреспондент	країна	світ	український	повідомлятися	перший	один
0	4	1	4	0	2	0	4	1	1	3
1	2	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	1
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0
5	1	0	1	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	1	0
8	0	0	0	1	0	0	0	2	0	0
9	0	0	0	1	0	0	1	3	1	0
10	0	1	0	0	0	0	0	1	0	0
11	0	0	0	1	0	0	0	0	0	0
12	0	0	0	1	0	0	0	1	0	0
13	0	0	0	0	0	0	0	0	0	0
14	1	0	0	1	0	1	0	1	0	0

Рис. 11 – Цей самий Bag of Word для слів, що найчастіше зустрічаються

Контрольні запитання

1. **Процес нормалізації текстових даних.**

До процесу нормалізації належить приведення тексту до єдиного регістру слів, очищення від знаків пунктуації та розмітки. Також до цього процесу можна віднести словесне написання чисел і розшифровування скорочень тощо.

2. **Відмінність стемінгу на лематизації.**

Відмінність полягає в тому, що під час стемізації ми приводимо слово до його кореня шляхом усунення придатків: суфікса, приставки та закінчення. Щодо лематизації, то там ми приводимо слово до смислової канонічної форми слова. Наприклад дієслово приведемо до інфінітиву, а іменники та прикметники приведемо до називного відмінку однини.

3. **Векторні моделі тексту.**

У найпростішому випадку векторна модель передбачає зіставлення кожному документу частотного спектра слів і вектора в лексичному просторі. У процесі пошуку частотний портрет запиту розглядається як вектор у тому самому просторі та за ступенем близькості (відстань або кут між векторами) визначаються найбільш релевантні документи.

У більш просунутих векторних моделях розмірність простору скорочується відкиданням найпоширеніших або слів, що рідко зустрічаються, збільшуючи тим самим відсоток значимості основних слів.

4. **Метрика TF-IDF.**

(TF – частота слова, IDF – обратна частота документа)

TF-IDF - статистичний показник, що використовується для оцінки важливості слів у контексті документа, що є частиною колекції документів чи корпусу. Його можна застосовувати як один з критеріїв релевантності документа до пошукового запиту, а також при розрахунку міри спорідненості документів при кластеризації.