قسم هندسة البرمجيات ونظم المعلومات

مشروع عملي نظم استرجاع المعلومات

السنة الخامسة

Information retrieval system

-search engine-

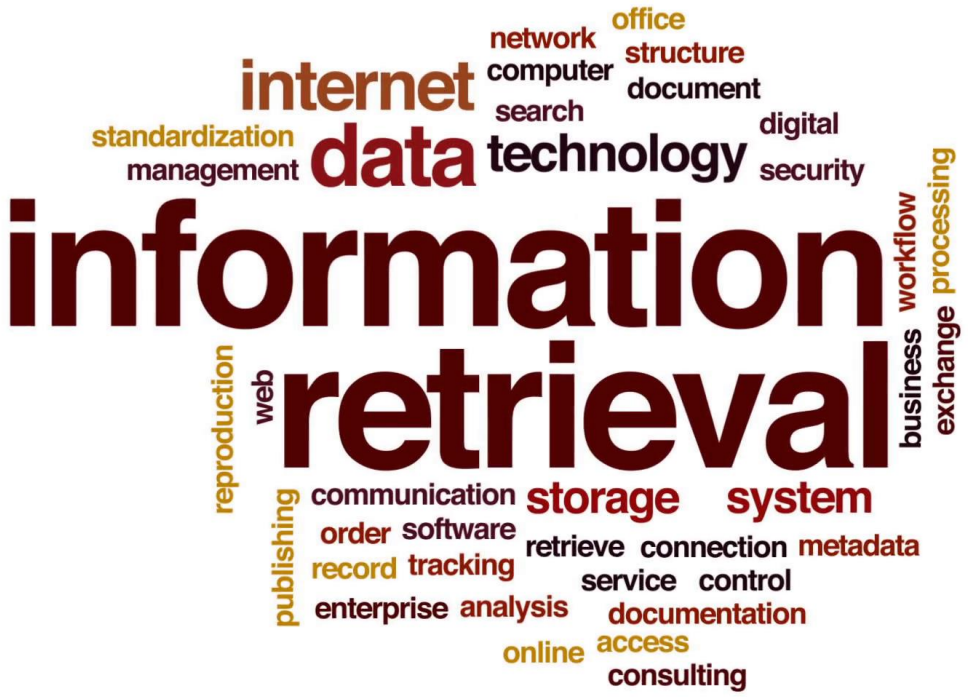
الطالبات:

آية عصفور بشرى صالح

تهاني عبدالهادي عليا سوار

المهندسة :

لين قويدر



توصيف الdata sets المستخدمة:

تم بناء النظام من خلال 2 data set وهما:

- Anatique Contains 404k docs and 5.0k queries
- Bear Contains 523k docs and 200k queries

حيث تحتوي كل منهما على مقالات يتم تمييز كل مقالة من خلال:

- Doc_id
- Text

وأيضاً كل استعلام في ملفات الاستعلام الموجودة في كل منهما مميزة من خلال:

- Query_id
- Text

وكلاهما يحتوي على ملف **qrels** لمعرفة مدى ترابط الملفات مع الاستعلامات لاستخدامها في التقييم

وتحوي على:

- Query_id
- Doc_id
- Relevance

وتم استخدام مكتبة **pandas** من أجل التعامل مع(الملفات ,الاستعلامات)

توصيف خطوات المشروع:

الطلب الأول:

في البداية قمنا بمعالجة نصية للملفات باستخدام مكتبة nltk حيث تم تطبيق:

- Remove stop words

```
#remove stop_
import nltk
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
df['text_copy'] = df['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
```

- remove punctuation

```
punctuation_regex=r'[!\"\$%&\'\(\)\*\+,-\./:;<=>\?\[\\\]\^`{\|}~،“”؟]'
#remove punctuation
def remove_punctuations(text):

    text = re.sub(punctuation_regex, ' ', str(text));
    return text
```

- Remove redundancy : إزالة الأحرف المكررة بشكل خاطئ

```
#delet_rep
def del_rep_letters(text):

    pattern = r"(.)\1{2,}"
    match = re.search(pattern,str(text))
    # re.sub(match.group(),"i",text)
    if(match):
        if(match.start()):
            return re.sub(re.escape(match.group()),text[match.start()]+text[match.start()+1],text)
    else:
        return text
```

- Remove none English word

```
import enchant
def remove_non_english_words(text):
    d = enchant.Dict("en_US")
    words = text.split()
    english_words = []

    tagged_words = nltk.pos_tag(words)

    for word, tag in tagged_words:
        if d.check(word) or is_proper_noun(tag):
            english_words.append(word)
        english_text = ' '.join(english_words)
    return english_text;
```

- **auto correct** باستخدام مكتبة **speller** على المصطلحات الانكليزية
- **Normalize** (**Date-Country**)

```
def is_date(string, fuzzy=False):
    try:
        parse(string,fuzzy=fuzzy)
        return True
    except ValueError:
        return False

def Convert(string):
    if is_date(string):
        string= parse(string).strftime("%d-%m-%Y")
    return string

def Date_normlization(text):
    return [Convert(w) for w in text]
from dateutil import parser

def is_country(term):
    for country in pycountry.countries:
        if country.alpha_2 == term.upper() or country.alpha_3 == term.upper():
            term=country.name
    return term
input_countries =['AS', 'CA', 'US',"DDH"]
def CountryNormlization(text):
    return [is_country(w) for w in text]
```

- **Tokenizing**:ليتم تقسيم كل سطر من الملفات الى كلمات مفهومة, ليسهل تحليلها بالإضافة بهذه الطريقة تم تحويل البيانات الغير مهيكلة الى بيانات منظمة.

```
# toknize _
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize, word_tokenize
import nltk
# nltk.download('book')
import re
df['text_copy']=df['text_copy'].apply(lambda x: word_tokenize(x))
df.head(10)
```

- **Lemmatizing**

تم استخدام Speech of Parts Tagging(sop) لتمييز أجزاء من الكلام ، حيث يمكننا محاولة وضع علامات على الكلمات قبل التجذير(lemmatizing) لتجنب خلط الكلمات المتجانسة ، أو الكلمات التي يتم تهجئتها بنفس الطريقة ولكن لها معاني مختلفة ويمكن أن تكون أجزاء مختلفة من الكلام ثم تم تطبيق ال lemmatize_textعلى كل من الword&tag

```
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer

# perform POS tagging
def lemmatize_text(words):
    #words = word_tokenize(words)
    pos_tags = nltk.pos_tag(words)


# create a lemmatizer object
    lemmatizer = WordNetLemmatizer()

# lemmatize the words based on their POS tags
    lemmatized_words = []
    for word, tag in pos_tags:
        if tag.startswith('NN'):
        # noun
            lemmatized_words.append(lemmatizer.lemmatize(word, pos='n'))
        elif tag.startswith('VB'):
        # verb
            lemmatized_words.append(lemmatizer.lemmatize(word, pos='v'))
        elif tag.startswith('JJ'):
        # adjective
            lemmatized_words.append(lemmatizer.lemmatize(word, pos='a'))
        elif tag.startswith('R'):
        # adverb
            lemmatized_words.append(lemmatizer.lemmatize(word, pos='r'))
        else:
        # other parts of speech
            lemmatized_words.append(word)
    return lemmatized_words
# print the lemmatized words
```

- **steaming**

```
#stemaization
from nltk.stem.snowball import SnowballStemmer
# Use English stemmer.
stemmer = SnowballStemmer("english")
df['text_copy'] = df['text_copy'].apply(lambda x: [stemmer.stem(y) for y in x])
```

الطلب الثاني:

```
In [11]: process(df)
         df.to_csv("c:/tmp/per_data.csv")
         #after process
```

```
In [10]: df = pd.read_csv('c:/tmp/per_data.csv')
         df
```

Out[10]:

	Unnamed: 0	doc_id	text	text_copy
0	0	1	What is the step by step guide to invest in sh...	['what', 'step', 'step', 'guid', 'invest', 'sh...
1	1	2	What is the step by step guide to invest in sh...	['what', 'step', 'step', 'guid', 'invest', 'sh...
2	2	3	What is the story of Kohinoor (Koh-i-Noor) Dia...	['what', 'stori', 'kohinoor', 'koh', 'i', 'noo...
3	3	4	What would happen if the Indian government sto...	['what', 'would', 'happen', 'indian', 'govern'...
4	4	5	How can I increase the speed of my internet co...	['how', 'i', 'increas', 'speed', 'internet', '...
...	...	...	...	...
522926	522926	537929	What's this coin?	['what', 's', 'coin']
522927	522927	537930	What is the approx annual cost of living while...	['what', 'approx', 'annual', 'cost', 'live', '...
522928	522928	537931	I am having little hairfall problem but I want...	['i', 'littl', 'hairfal', 'problem', 'i', 'wan...
522929	522929	537932	What is like to have sex with cousin?	['what', 'like', 'sex', 'cousin']
522930	522930	537933	What is it like to have sex with your cousin?	['what', 'like', 'sex', 'cousin']

522931 rows × 4 columns

```
In [3]: dataset = ir_datasets.load("beir/quora/dev")
        df=pd.DataFrame(dataset.docs)
        #print df befor process
        df
```

Out[3]:

	doc_id	text
0	1	What is the step by step guide to invest in sh...
1	2	What is the step by step guide to invest in sh...
2	3	What is the story of Kohinoor (Koh-i-Noor) Dia...
3	4	What would happen if the Indian government sto...
4	5	How can I increase the speed of my internet co...
...	...	...
522926	537929	What's this coin?
522927	537930	What is the approx annual cost of living while...
522928	537931	I am having little hairfall problem but I want...
522929	537932	What is like to have sex with cousin?
522930	537933	What is it like to have sex with your cousin?

522931 rows × 2 columns

الطلب الثالث:

لبناء الفهرس قمنا ببناء Vector Space Model

قمنا بداية بحساب TF-IDF لكل Term و Document باستخدام sklearn.feature_extraction.text

ومن ثم معرفة كل ال docs التي تواجد فيها كل termمعين عن طريق الinverted index

```
def Tf_idfData(df):
    vectorizer = TfidfVectorizer(tokenizer=dummy, preprocessor=dummy)
    X = vectorizer.fit_transform(df['text_copy'].values)
    sparse_matrix = csr_matrix(X)
    df1 = pd.DataFrame.sparse.from_spmatrix(sparse_matrix, columns=vectorizer.get_feature_names())
    return df1,X,vectorizer
def inverted_index(df1):
    inv_indx = defaultdict(list)
    for term in df1.columns:
        indices= df1[df1[term] > 0].index.tolist()
        inv_indx[term]=indices
    for term , indices in inv_indx:
        print(f"{indices}:{term}")
    df1.to_csv("inverted_index.csv")
    return df1
```

الطلب الرابع:

تمت معالجة الاستعلامات بنفس الطريقة التي تم فيها معالجة الملفات في الطلب الأول

	text_copy	text	query_id
0	[how, quora, look, moder]	?How does Quora look to a moderator	318
1	[how, i, refus, chose, differ, thing, life]	...How do I refuse to chose between different thi	378
2	...did, ben, affleck, shine, christian, bale, ba]	...Did Ben Affleck shine more than Christian Bale	379
3	... ,what, effect, demonit, 500, 100, rupe, note]	... What are the effects of demonitization of 500	399
4	[whi, creativ, import]	?Why creativity is important	420
...	...	...	...
4995	[how, make, instrument]	?How do you make instrumentals	537329
4996	[how, meo, class, 4, exam]	?How will be MEO Class 4 exam	537374
4997	[how, i, remov, hesit]	?How I can remove my hesitation	537736
4998	[whi, transit, metal, use, catalyst]	?Why are transition metals used as catalysts	537788
4999	[what, interest, book, side, atheism]	...What are the most interesting books on the sid	537790

rows × 3 columns 5000

الطلب الخامس:

يتم نمذجة المستندات كـ **vectors**،كما يتم نمذجة الاستعلام أيضا كـ **vector** بالتالي للمطابقة بينهم نحتاج لتابع مسافة مثل ال **similarity cosine** حيث نحصل على المسافة من خالل الزاوية فـال **cos** ألكبر هو صاحب الزاوية ألصغر ومنه نستطيع ترتيب كامل االشعة حسب ال **cos** فنحصل على ترتيب كل الـ **documents** بحسب ألقرب للـ **query** و قمنا بذلك من خلال تابع **def most_similar**

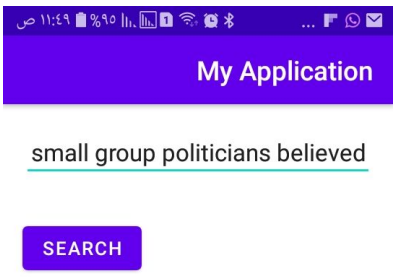
```
que1="Why do some men spit into the urinal before urinating?"
quer=Query_processing(que1)
print ("processed query",quer)
print (quer)
search = vectorizer.transform([quer])
df2 = pd.DataFrame(search.toarray(), columns=vectorizer.get_feature_names())
print(df2)
from sklearn.metrics.pairwise import cosine_similarity,cosine_distances

def cos_similarity(search_query_weights, tfidf_weights_matrix):
    cosine_distance = cosine_similarity(search_query_weights, tfidf_weights_matrix)
    similarity_list = cosine_distance[0]

    return similarity_list
def most_similar(similarity_list, min_Doc=0):
    most_similar = []
    most_similar2 = []
    for i in range(0, len(similarity_list)):
        if similarity_list[i] > 0.0:
            tmp_index = np.argmax(similarity_list)
            most_similar.append(tmp_index)
            similarity_list[tmp_index] = 0

    return most_similar
```


الطلب السادس:



الطلب الإضافي:

Crawling:

حيث تم استخراج الروابط الموجودة داخل datasets وإضافة المعلومات الموجودة ضمن الروابط إلى الdataالموجودة سابقا ضمن الdata sets

```
        for paragraph in paragraphs:
            data = data.append({ 'text': paragraph.text}, ignore_index=True)
        return data
```

In [31]:
```
from urlextract import URLExtract
import numpy as np
import requests
from bs4 import BeautifulSoup
extractor = URLExtract()
da=pd.DataFrame()
import re
URLPATTERN = r'(https?://\S+)'

da['url'] = df['text'][:1000].apply(lambda x: re.findall(URLPATTERN, x))
#da['url'] =  da['url'].apply(lambda x: "".join(x))
list_url = sum(da['url'],[])
```

In [33]:
```
for t in range(0,list_url):
    data=process_url(list_url[t])
    data.to_csv("c:/tmp/cours.csv', sep = '|', mode='a', index= False,header=False)
```

In [34]:
```
da = pd.read_csv('c:/tmp/cours.csv')
da
```

Out[34]:

	This website is using a security service to protect itself from online attacks. The action you just performed triggered the security solution. There are several actions that could trigger this block including submitting a certain word or phrase	a SQL command or malformed data.
0	You can email the site owner to let them know ...	NaN
1	\nCloudflare Ray ID: 7d42428a1bd4badf\n'\n\n ...	NaN
2	This website is using a security service to pr...	a SQL command or malformed data.

```
dataset = ir_datasets.load("antique/train/split200-train")
df=pd.DataFrame(dataset.docs)
```

In [24]:
```
def process_url(url):
    if url:
        response = requests.get(url)
        soup = BeautifulSoup(response.content)
    # Find all links on the page
        links = soup.find_all('h')
    # Find all paragraphs on the page
        paragraphs = soup.find_all('p')
    # Create a DataFrame to store the data
        data = pd.DataFrame(columns=['text'])

    # Iterate over the links and paragraphs and store the data in the DataFrame
        for link in links:
            data = data.append({'text': link.text}, ignore_index=True)
        for paragraph in paragraphs:
            data = data.append({ 'text': paragraph.text}, ignore_index=True)
        return data
```

In [25]:
```
from urlextract import URLExtract
import numpy as np
import requests
from bs4 import BeautifulSoup
extractor = URLExtract()
da=pd.DataFrame()
import re
URLPATTERN = r'(https?://\S+)'

da['url'] = df['text'][:180].apply(lambda x: re.findall(URLPATTERN, x))
#da['url'] =  da['url'].apply(lambda x: "".join(x))
list_url = sum(da['url'],[])
```

توصيف بنية النظام architecture system وكيفية التواصل بين الخدمات:

تمت القراءة بداية من الملف ثم تم تخزين نتيجة القراءة ضمن متحول وتمت معالجة نصية لهذا المتحول ثم تمت عملية التجذير(stemming) على النص لاعطاء نتائج دقيقة وقمنا بعمل (lemmatization)

تمثيل المستندات كvectorبحيث يتم أيضا تمثيل الاستعلام كشعاع ليتم حساب التشابه فيما بينهم واستعمال مكتبة pandas

النتائج والتقييم على الdata set :

سنطبق مجموعة من المقاييس على نظامنا لنعلم مدى صحة نتائج هذا النظام ومدى مطابقتها للنتائج الصحيحة التي ستظهر عند الاستعلامات

تم تطبيق evaluation على 2000 استعلام

ملاحظة:

تم تطبيق precision@10 على retrieved doc=10 أما الباقي فتم تطبيقهم على جميع retrieved doc التي تم إرجاعهم

```
1997    query_id: 196563        relevant: 2     true_Positive: 0        false 10
1998    query_id: 196683        relevant: 1     true_Positive: 0        false 10
1999    query_id: 196757        relevant: 1     true_Positive: 0        false 10
2000    query_id: 196811        relevant: 1     true_Positive: 0        false 10
2001    query_id: 196964        relevant: 1     true_Positive: 0        false 10
2002    query_id: 197291        relevant: 1     true_Positive: 0        false 10
2003    query_id: 197406        relevant: 1     true_Positive: 0        false 10
2004    query_id: 197493        relevant: 1     true_Positive: 0        false 10
2005    query_id: 197624        relevant: 1     true_Positive: 0        false 10
2006    query_id: 197820        relevant: 6     true_Positive: 0        false 10
2007    query_id: 197876        relevant: 4     true_Positive: 0        false 10
2008    query_id: 198039        relevant: 2     true_Positive: 0        false 10
2009    query_id: 198138        relevant: 1     true_Positive: 0        false 10
2010    query_id: 198172        relevant: 1     true_Positive: 0        false 10
```

```
In [51]: maap=sum(average_precision) / 2000
         print("map is:",maap)
         mrr_score = sum(reciprocal_ranks) / 2000
         print("Mrr is:",mrr_score)
         average_precision = sum(precision_list)
         average_recall = sum(recall_list)
         print("Average Precision is : ", average_precision)
         print("Average Recall is : ", average_recall)

         map is: 3.0629676051461565e-05
         Mrr is: 0.08348700791127264
         Average Precision is :  0.2607563834698251
         Average Recall is :  1403.0206089988328
```

```
1       query_id: 378   relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
2       query_id: 379   relevant: 5     true_Positive: 4        false 6
        precision@10 0.4        recall 0.8
3       query_id: 399   relevant: 28    true_Positive: 7        false 3
        precision@10 0.7        recall 0.25
4       query_id: 420   relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
5       query_id: 540   relevant: 2     true_Positive: 1        false 9
        precision@10 0.1        recall 0.5
6       query_id: 548   relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
7       query_id: 609   relevant: 2     true_Positive: 1        false 9
        precision@10 0.1        recall 0.5
8       query_id: 744   relevant: 10    true_Positive: 1        false 9
        precision@10 0.1        recall 0.1
9       query_id: 784   relevant: 2     true_Positive: 2        false 8
        precision@10 0.2        recall 1.0
10      query_id: 858   relevant: 2     true_Positive: 1        false 9
        precision@10 0.1        recall 0.5
11      query_id: 975   relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
12      query_id: 1079  relevant: 2     true_Positive: 0        false 10
13      query_id: 1088  relevant: 2     true_Positive: 0        false 10
14      query_id: 1164  relevant: 7     true_Positive: 0        false 10
15      query_id: 1166  relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
16      query_id: 1248  relevant: 16    true_Positive: 8        false 2
        precision@10 0.8        recall 0.5
17      query_id: 1350  relevant: 2     true_Positive: 1        false 9
        precision@10 0.1        recall 0.5
18      query_id: 1453  relevant: 3     true_Positive: 1        false 9
        precision@10 0.1        recall 0.3333333333333333
19      query_id: 1578  relevant: 1     true_Positive: 1        false 9
        precision@10 0.1        recall 1.0
20      query_id: 1803  relevant: 1     true_Positive: 0        false 10
21      query_id: 1956  relevant: 6     true_Positive: 0        false 10
22      query_id: 1992  relevant: 7     true_Positive: 1        false 9
        precision@10 0.1        recall 0.14285714285714285
```

تقسيم العمل:

الطلب الأول والثالث: بشرى صالح -عليا سوار

باقي الطلبات: آية عصفور -تهاني عبدالهادي

المصادر:

- https://www.pinecone.io/learn/offline-evaluation/
- **محاضرات العملي ومحاضرات الدكتور أبي صندوق**
- https://realpython.com/nltk-nlp-python/#using-named-entity-recognition-ner
- **https://wellsr.com/python/python-named-entity-recognition-with-nltk-and-spacy.2**