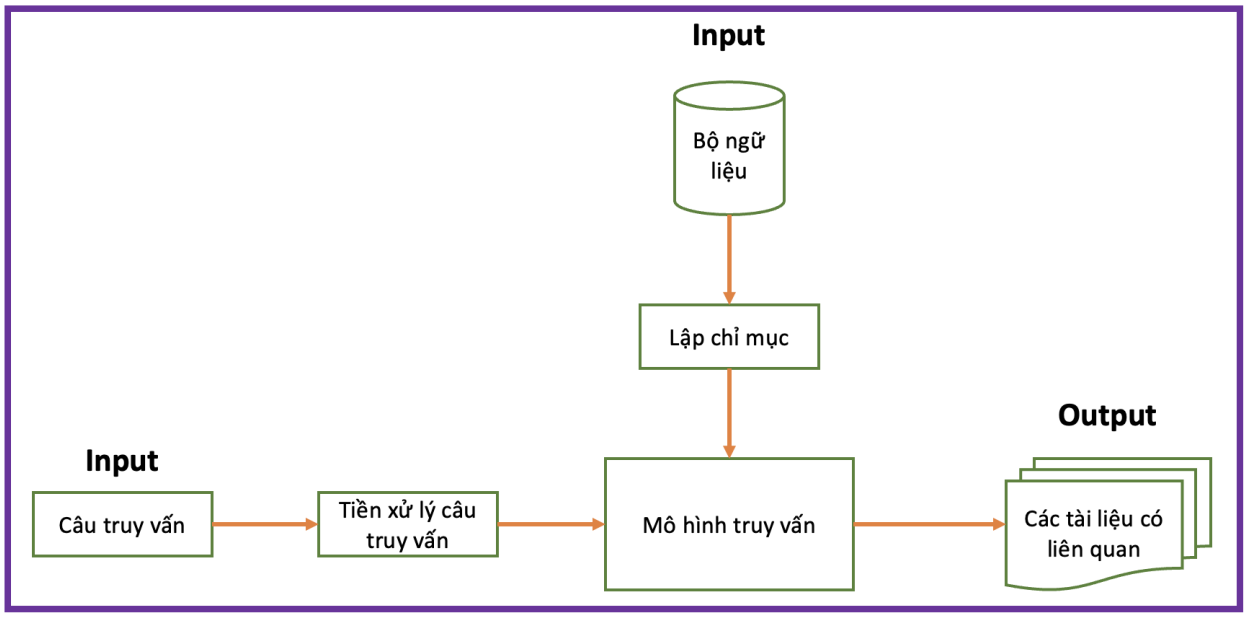
AI VIET NAM – COURSE 2023

Text Retrieval – Project

Ngày 9 tháng 7 năm 2023

Phần I: Mô tả project

Text Retrieval (Tạm dịch: Truy vấn văn bản): là một bài toán thuộc lĩnh vực Truy vấn thông tin (Information Retrieval). Trong đó, nhiệm vụ của ta là xây dựng một chương trình trả về các tài liệu (Document) có liên quan đến câu truy vấn (Query) đầu vào và các tài liệu được lấy từ một bộ ngữ liệu (Corpus) cho trước, trong lĩnh vực xử lý ngôn ngữ tự nhiên, bộ ngữ liệu có thể hiểu là một bộ dữ liệu văn bản hay tập các tài liệu văn bản. Có rất nhiều cách thiết kế hệ thống truy vấn văn bản khác nhau, tuy nhiên về mặt tổng quát sẽ có một pipeline chung sau đây:



Hình 1: Pipeline tổng quan của một hệ thống Text Retrieval.

Dựa vào hình trên, có thể phát biểu Input/Output của một hệ thống truy vấn văn bản bao gồm:

- **Input:** Câu truy vấn $\mathbf{q}$ và bộ ngữ liệu $\mathbf{C}$.
- **Output:** Danh sách các tài liệu $\mathbf{c}$ ($\mathbf{c} \in \mathbf{C}$) có nội dung liên quan đến câu truy vấn.

Hình 2: Demo về hệ thống truy vấn văn bản

Trong project này, chúng ta sẽ xây dựng một chương trình truy vấn văn bản với bộ dữ liệu **MS MARCO (Microsoft Machine Reading Comprehension Dataset)**, một bổ dữ liệu phục vụ cho các bài toán hỏi-đáp (Question Answering) với hơn 100,000 câu hỏi và câu trả lời tương ứng, qua thời gian đã phát triển lên cho các bài toán truy vấn văn bản. Các code triển khai dưới đây đều được thực hiện trong Google Colab.

1. **Tải bộ dữ liệu:** Đầu tiên, ta tải bộ dữ liệu MS MACRO sử dụng thư viện datasets bằng lệnh sau (các bạn có thể đọc thêm chi tiết về thư viện này tại đây):

```
!pip install datasets==2.13.1
```

Sau đó, ta tải bộ dữ liệu MS MACRO về thông qua lệnh sau:

```
from datasets import load_dataset

dataset = load_dataset('ms_marco', 'v1.1')
```

Dataset Viewer — Auto-converted to Parquet — API

Subset: v1.1 (102k rows) — Split: test (9.65k rows)

answers (sequence)	passages (sequence)	query (string)	query_id (int32)	query_type (string)	wellFormedAnswers (sequence)
["Yes"]	{ "is_selected": [0, 0, 1, 0, 0, 0, 0], "passage_text": ["We have been feeding our back…	"does human hair stop squirrels"	0	"description"	[]
["Fossil fuels are basically the remains of animals and plants and these are good…	{ "is_selected": [0, 1, 0, 0, 0, 0, 0, 0, 0], "passage_text": ["The biggest advantage of using…	"what are the benefits of fossil fuels"	1	"description"	[]
["The apothem of a regular polygon is a line segment from the center to the…	{ "is_selected": [0, 0, 0, 0, 0, 1, 0, 0, 0], "passage_text": ["Apothem. The apothem of a…	"what is a apothem"	2	"description"	[]
["$45 to $210. 2"]	{ "is_selected": [0, 0, 0, 0, 0, 1, 0, 0, 0], "passage_text": ["Congratulations! You have foun…	"average cost for custom canopy"	3	"numeric"	[]
["It is the collection of physical elements that constitutes a computer…	{ "is_selected": [0, 1, 0, 0, 0, 0, 0, 0, 0, 0], "passage_text": ["Hardware is best described as …	"what is a hardware in a computer"	4	"description"	[]
["A standard format for exchanging business data between organisations by…	{ "is_selected": [0, 0, 0, 0, 0, 1, 0, 1, 0, 0], "passage_text": ["Like many other early…	"edi logistics definition"	5	"description"	[]
["Marijuana should not be made legal for recreational use because it will have…	{ "is_selected": [0, 0, 0, 0, 0, 0, 0, 1, 0], "passage_text": ["Some people ask 'why should…	"why should recreational marijuana be illegal"	6	"description"	[]
["Arachnida"]	{ "is_selected": [0, 0, 0, 0, 1, 0], "passage_text": ["SUBPHYLUM CHELICERATA, CLASS…	"what class are spiders in"	7	"entity"	[]

Hình 3: Các trường thông tin có trong bộ dữ liệu MS MACRO.

Trong phạm vi của project, chúng ta chỉ quan tâm đến hai cột chính là:

- **"query (string)":** Cột chứa mô tả câu truy vấn.

- **"passages (sequence)":** Cột chứa các tài liệu có nội dung tương đồng câu truy vấn. Trong đó, tài liệu được gán $is\_selected = 1$ là nhãn của tài liệu có liên quan đến câu truy vấn (theo ngữ cảnh của bộ dữ liệu hỏi-đáp thì tài liệu này sẽ chứa câu trả lời cho câu hỏi (câu truy vấn)).

2. **Xây dựng danh sách câu truy vấn và tài liệu:** Với bộ dữ liệu hỏi-đáp vừa tải về, chúng ta sẽ thực hiện tách các trường thông tin cần thiết cho bài toán truy vấn văn bản gồm danh sách các câu truy vấn và tài liệu có liên quan tương ứng. Vì bộ dữ liệu này rất lớn, việc xử lý toàn bộ sử dụng Python sẽ rất tốn thời gian với. Vậy nên, chúng ta sẽ giới hạn phạm vi thực hiện trên tập **test** của bộ dữ liệu, đồng thời chỉ chọn các câu truy vấn thuộc kiểu $query\_type = "entity"$. Như vậy, các bước thực hiện như sau:

   (a) **Chọn bộ test:**

```
subset = dataset['test']
```

   (b) **Khai báo danh sách chứa tập câu truy vấn và tài liệu có liên quan:**

```
queries_infos = []
queries = []
corpus = []
```

   (c) **Thực hiện việc tách dữ liệu:**

```
for sample in subset:
    query_type = sample['query_type']
    if query_type != 'entity':
        continue
    query_id = sample['query_id']
    query_str = sample['query']
    passages_dict = sample['passages']
    is_selected_lst = passages_dict['is_selected']
    passage_text_lst = passages_dict['passage_text']
    query_info = {
        'query_id': query_id,
        'query': query_str,
        'relevant_docs': []
    }
    current_len_corpus = len(corpus)
    for idx in range(len(is_selected_lst)):
        if is_selected_lst[idx] == 1:
            doc_idx = current_len_corpus + idx
            query_info['relevant_docs'].append(doc_idx)

    if query_info['relevant_docs'] == []:
        continue

    queries.append(query_str)
    queries_infos.append(query_info)
    corpus += passage_text_lst
```

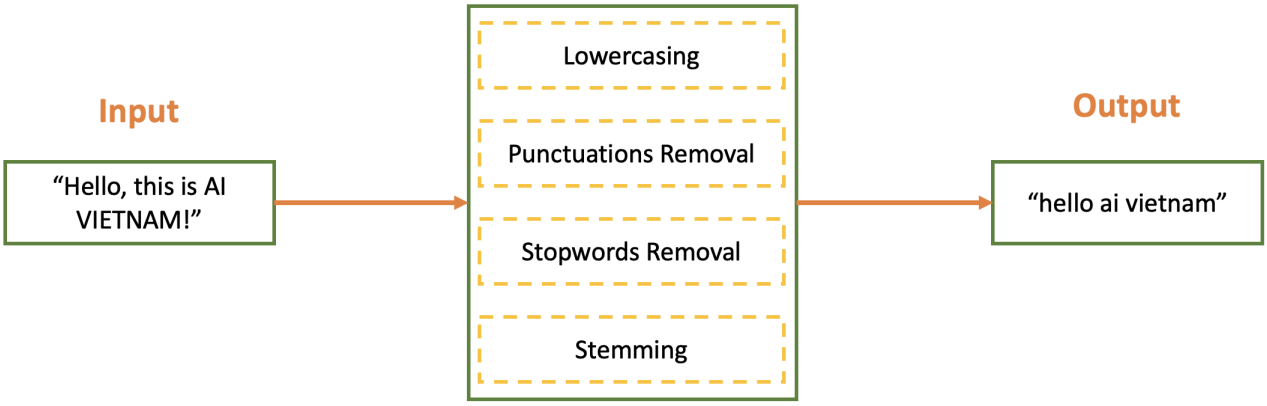
Trong đó:

- **Dòng 1, 2, 3, 4:** Duyệt qua từng sample trong bộ test và chỉ chọn các sample có câu truy vấn thuộc kiểu **"entity"**.
- **Dòng 5, 6:** Lấy nội dung câu truy vấn và id của chúng (các bạn có thể bỏ qua phần id này để giảm tải bộ nhớ).
- **Dòng 7, 8, 9:** Lấy danh sách các tài liệu và nhãn tương ứng của câu truy vấn.
- **Dòng 10, 11, 12, 13, 14:** Khai báo dict chứa thông tin cho câu truy vấn.

- **Dòng 15, 16, 17, 18, 19:** Từ danh sách các tài liệu và nhãn, ta chọn các tài liệu được gán có liên quan đến câu truy vấn và đưa vào key **"relevant_docs"**. Ta lưu trữ nhãn dưới dạng chỉ mục trong list **corpus**.
- **Dòng 21, 22:** Ta bỏ qua các sample không chứa tài liệu có liên quan cho câu truy vấn để thuận tiện trong việc đánh giá.
- **Dòng 24, 25, 26:** Đưa thông tin câu truy vấn và tài liệu vào các danh sách đã khai báo ở bước 2.

3. **Xây dựng hàm chuẩn hóa văn bản:** Với hầu hết các bài toán liên quan đến văn bản, việc chuẩn hóa văn bản là vô cùng quan trọng bởi nó giúp ta phần nào giảm được độ phức tạp trong việc biểu diễn văn bản. Trong project này, các bạn sẽ xây dựng một hàm **text_normalize()** với tham số đầu vào là một chuỗi **s**, sau đó trả về một chuỗi đã được chuẩn hóa. Các kỹ thuật ta dùng trong project này bao gồm:

   - **Chuyển chữ viết thường (Lowercasing)**
   - **Xóa dấu câu (Punctuations Removal)**
   - **Xóa stopwords (Stopwords Removal)**
   - **Stemming**



Hình 4: Input/Output của hàm chuẩn hóa văn bản

Các bạn có thể xây dựng hàm chuẩn hóa văn bản theo cách sau:

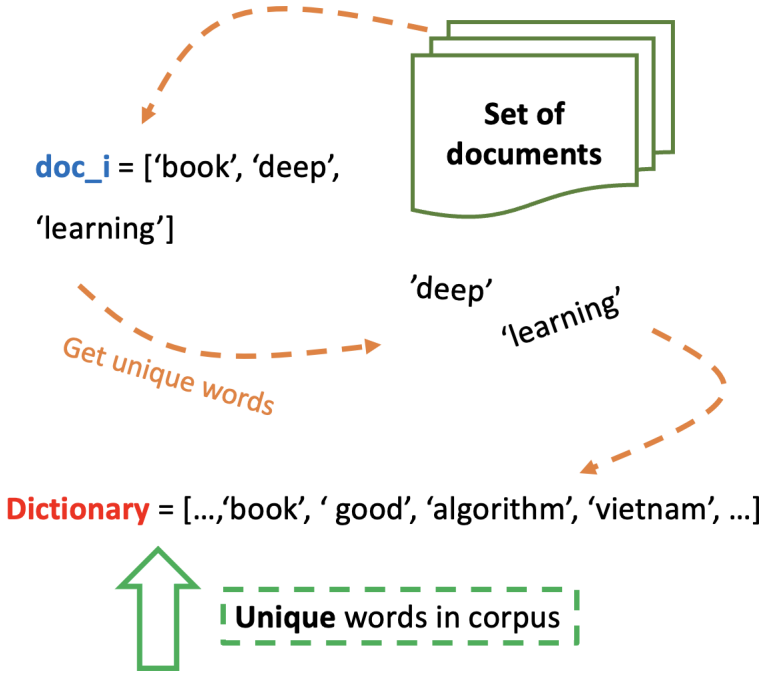
```
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

nltk.download('stopwords')
english_stopwords = stopwords.words('english')
remove_chars = string.punctuation
stemmer = PorterStemmer()

def text_normalize(text):
    text = text.lower()
    for char in remove_chars:
        text = text.replace(char, '')
```

```
15      text = ' '.join([word for word in tokenize(text) if word not in
        english_stopwords])
16      text = ' '.join([stemmer.stem(word) for word in tokenize(text)])
17
18      return text
```

Trong đó:

- **Dòng 1:** Import module string trong Python.
- **Dòng 2, 3, 4:** Import thư viện nltk (thư viện này đã được cài đặt sẵn trong Google Colab) và import 2 module về stopwords và stemming của thư viên này.
- **Dòng 6, 7:** Tải bộ stopwords của tiếng Anh về và khai báo thành list.
- **Dòng 8:** Khai báo danh sách các dấu câu cần xóa (danh sách này dưới dạng string).
- **Dòng 9:** Khai báo một thực thể PorterStemmer, dùng để gọi hàm stemming về sau.
- **Dòng 11:** Khai báo hàm text_normalize() với tham số đầu vào là một string.
- **Dòng 12:** Thực hiện kỹ thuật chuyển chữ viết thường.
- **Dòng 13, 14:** Thực hiện kỹ thuật xóa dấu câu.
- **Dòng 15:** Thực hiện kỹ thuật xóa stopwords.
- **Dòng 16:** Thực hiện kỹ thuật stemming.

4. **Xây dựng bộ từ vựng (dictionary):** Từ bộ ngữ liệu cho trước, ta duyệt qua nội dung của từng tài liệu, thực hiện chuẩn hóa văn bản và sử dụng kỹ thuật Tokenization để tách thành các token. Với mỗi token nhận được, kiểm tra nếu token hiện tại không tồn tại trong bộ từ vựng thì ta sẽ thêm nó vào.



Hình 5: Ảnh minh họa quá trình xây dựng bộ từ vựng.

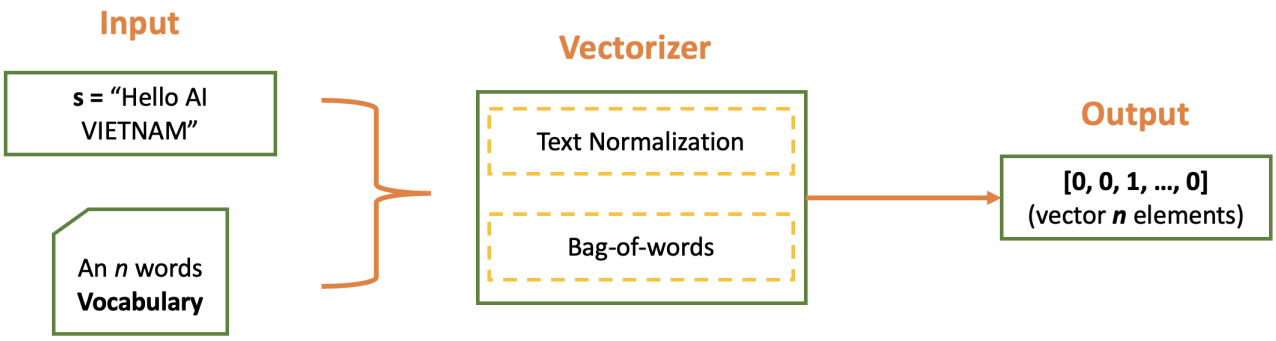
Theo đó, ta sẽ triển khai hàm xây dựng theo cách như sau:

```
1  def create_dictionary(corpus):
2      dictionary = []
3      for doc in corpus:
4          normalized_doc = text_normalize(doc)
5          tokens = tokenize(normalized_doc)
6          for token in tokens:
7              if token not in dictionary:
8                  dictionary.append(token)
9
10     return dictionary
```

Trong đó:

- **Dòng 1:** Khai báo hàm **create_dictionary()** nhận đầu vào là danh sách các tài liệu (string).
- **Dòng 2:** Khai báo một list rỗng, dùng để chứa các từ vựng về sau.
- **Dòng 3, 4:** Duyệt qua các tài liệu trong corpus và thực hiện chuẩn hóa chúng.
- **Dòng 5:** Thực hiện tách token các tài liệu đã được chuẩn hóa.
- **Dòng 6, 7, 8:** Duyệt qua từng token đã tách, nếu token nào chưa có trong bộ từ vựng thì thêm vào.
- **Dòng 10:** Trả về list dictionary đã khai báo ở dòng 2.

5. **Xây dựng hàm vector hóa văn bản:** Vì máy tính không thể sử dụng chuỗi văn bản để thực hiện tính toán, ta cần biểu diễn lại các văn bản dưới dạng vector. Trong project này, ta xây dựng một hàm **vectorize()** với tham số đầu vào là một chuỗi văn bản **s** và bộ từ vựng **vocab**, sau đó thực hiện vector hóa văn bản sử dụng kỹ thuật bag-of-words (các bạn có thể tham khảo lý thuyết về bag-of-words tại đây) dựa trên bộ từ vựng cho trước, kết quả trả về sẽ là vector bag-of-words của văn bản đầu vào.



Hình 6: Input/Output của hàm vector hóa văn bản. Hàm này sẽ thực hiện chuẩn hóa văn bản sau đó tính bag-of-words với bộ từ vựng có sẵn.

Các bạn có thể triển khai code cho hàm vector hóa văn bản như sau:

```
def vectorize(text, dictionary):
    word_count_dict = {word: 0 for word in dictionary}
    tokens = tokenize(text)
    for token in tokens:
        try:
            word_count_dict[token] += 1
        except:
            pass
```

```
9
10      vector = list(word_count_dict.values())
11
12      return vector
```

Trong đó:

- **Dòng 1:** Khai báo hàm **vectorize()** nhận tham số đầu vào là một văn bản kiểu string và một bộ từ vựng kiểu list.
- **Dòng 2:** Xây dựng một dictionary dùng để tạm thời khởi tạo giá trị 0 cho toàn bộ các từ vựng trong bộ từ vựng.
- **Dòng 3:** Thực hiện tách token văn bản đầu vào.
- **Dòng 4, 5, 6, 7, 8:** Duyệt qua danh sách token, nếu token có trong bộ từ vựng thì sẽ +1, nếu không sẽ bỏ qua.
- **Dòng 10, 12:** Lấy danh sách các value trong dictionary ở dòng 2 để làm vector đại diện cho văn bản đầu vào và trả về.

Lưu ý: ở bản cài đặt này ta sẽ không thực hiện chuẩn hóa văn bản bên trong hàm **vectorize()** mà sẽ thực hiện khi ta bắt đầu tiến hành truy vấn ở phần cuố. Vì vậy về mặt lý thuyết, các bước làm vẫn không đổi.

6. **Xây dựng ma trận document-term:** Với các tài liệu trong bộ ngữ liệu, ta cần lưu trữ dạng biểu diễn vector của chúng trong một cấu trúc được gọi là ma trận document-term. Trong ma trận này, mỗi hàng sẽ đại diện cho một tài liệu và mỗi cột sẽ đại diện cho mỗi từ (term) trong bộ từ vựng. Như vậy, giả sử với một bộ ngữ liệu gồm có ba tài liệu văn bản có nội dung như sau:

   - **doc1** = "Read AI book"
   - **doc2** = "Machine Learning book"
   - **doc3** = "Learning how AI learns"

   Sau khi thực hiện chuẩn hóa văn bản và tokenization, ta thu thập các token độc nhất trong toàn bộ bộ ngữ liệu để lập thành bộ từ vựng gồm ['read', 'book', 'ai', 'machine', 'learn', 'how']. Dựa vào danh sách từ vựng có được, tính tần suất xuất hiện của mỗi từ trong bộ từ vựng với từng văn bản đã được tokenize để từ đó có được dạng vector biểu diễn của chúng. Trong ma trận document-term, ba vector này chính là 3 hàng của ma trận tượng trưng cho 3 tài liệu và các giá trị trong vector sẽ tượng trưng cho các cột đại diện cho các từ trong bộ từ vựng.

doc1 = "Read AI book" →
doc2 = "Machine Learning book" →
doc3 = "Learning how AI learns" →

	read	book	ai	machine	learn	how
doc1	1	1	1	0	0	0
doc2	0	1	0	1	1	0
doc3	0	0	1	0	2	1

Hình 7: Ma trận document-term. Mỗi hàng đại diện cho chỉ mục mỗi tài liệu và mỗi cột đại diện cho các term trong bộ từng vựng.

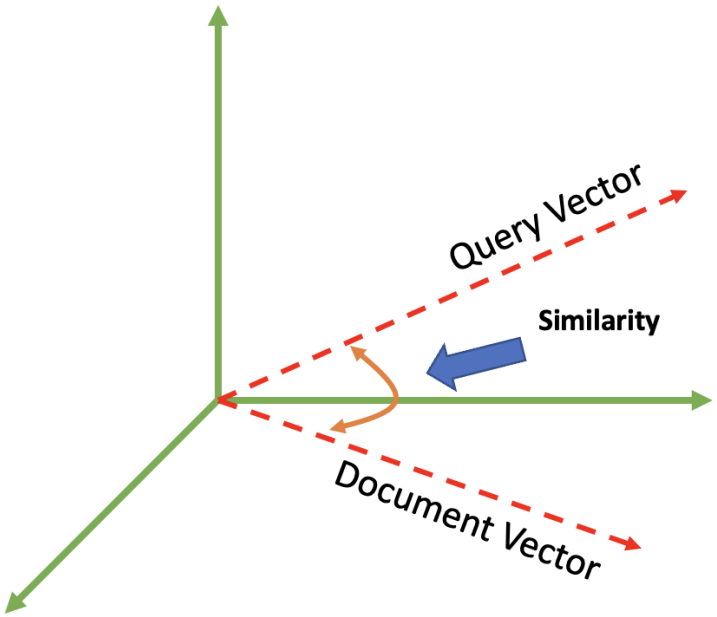
Để xây dựng ma trận doc-term, các bạn có thể triển khai như sau:

```
1 def create_doc_term_matrix(corpus, dictionary):
2     doc_term_matrix = {}
3     for idx, doc in enumerate(corpus):
4         normalized_doc = text_normalize(doc)
5         vector = vectorize(normalized_doc, dictionary)
6         doc_term_matrix[(doc, idx)] = vector
7 
8     return doc_term_matrix
```

Trong đó:

- **Dòng 1:** Khai báo hàm **create_doc_term_matrix()** nhận tham số đầu vào là list các tài liệu và list các từ vựng.
- **Dòng 2:** Khai báo một dict rỗng dùng để làm ma trận doc-term.
- **Dòng 3, 4, 5, 6:** Duyệt qua từng tài liệu trong bộ ngữ liệu, thực hiện vector hóa văn bản và thêm vào ma trận doc-term. Để demo được rõ ràng, ta đưa cả nội dung tài liệu vào phần key. Chỉ mục của tài liệu cũng được đưa vào để thuận tiện trong việc đánh giá.
- **Dòng 8:** Trả về ma trận doc-term.

7. **Xây dựng hàm tính độ tương đồng giữa hai vector:** Để có thể tìm được các tài liệu có liên quan đến câu truy vấn, ta có thể sử dụng các công thức được dùng để đo sự tương đồng giữa hai vector (ở đây sẽ hiểu là vector tài liệu và vector câu truy vấn), từ đó xây dựng một hàm **similarity()** nhận đầu vào là hai vector có cùng kích thước, sau đó trả về một giá trị là điểm đại diện cho độ tương đồng của vector này.



Hình 8: Độ tương đồng giữa hai vector câu truy vấn và tài liệu theo phép cosine similarity.

Trong project này, ta sẽ dùng độ đo Cosine Similarity để tính độ tương đồng giữa 2 vector, công thức như sau:

$$cosine\_similarity(\vec{a}, \vec{b}) = \frac{a \cdot b}{\|a\|\|b\|} = \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{N} a_i^2}\sqrt{\sum_{i=1}^{N} b_i^2}}$$

Các bạn có thể sử dụng thư viện **scipy** để triển khai cosine similarity:

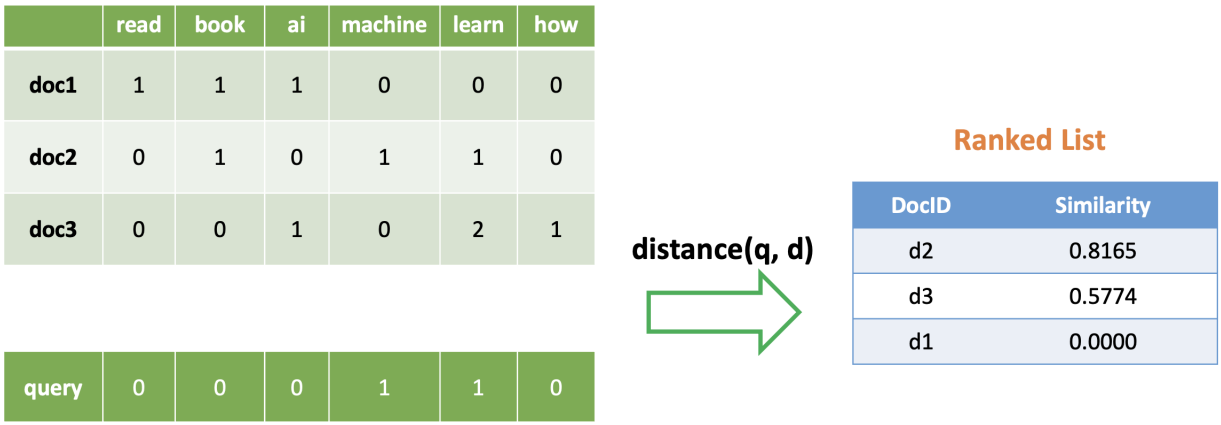
```
from scipy import spatial

def similarity(a, b):
    return 1 - spatial.distance.cosine(a, b)
```

8. **Xây dựng hàm truy vấn** Cuối cùng, dựa trên các thành phần đã xây dựng ở các bước trên, ta sẽ bắt đầu thực hiện tìm kiếm các tài liệu có liên quan đến một chuỗi văn bản câu truy vấn cho trước, các bước thực hiện như sau:

   (a) **Vector hóa câu truy vấn:** Với một chuỗi văn bản truy vấn cho trước, sử dụng hàm **vectorize()** đã khai báo trước đó để tìm được dạng biểu vector của câu truy vấn. Giả sử, với câu truy vấn có nội dung **q = "Machine Learning"**, khi đưa vào **vectorize()**, chuỗi câu truy vấn sẽ được chuẩn hóa và tính vector bag-of-words trên bộ vocab đã tìm được trước đó, cuối cùng sẽ được kết quả là {'read': 0, 'book': 0, 'ai': 0, 'machine': 1, 'learn': 1, 'how': 0} hay $[0, 0, 0, 1, 1, 0]$.

   (b) **Tính độ tương đồng:** Với từng tài liệu trong ma trận document-term, thực hiện tính độ tương đồng giữa vector câu truy vấn với vector tài liệu (có thể lưu kết quả này vào một list).

   (c) **Thực hiện xếp hạng:** Sau khi có danh sách các điểm tương đồng, thực hiện sắp xếp theo thứ tự giảm dần về điểm tương đồng. Cuối cùng, các tài liệu nằm ở đầu danh sách đã sắp xếp này sẽ có thể được coi là có liên quan nhất đến câu truy vấn.



Hình 9: Thực hiện truy vấn và lập bảng xếp hạng các tài liệu có liên quan theo điểm độ tương đồng tính được.

Ta sẽ gói các bước trên thành một hàm duy nhất như sau:

```
def ranking(query, dictionary, doc_term_matrix):
    normalized_query = text_normalize(query)
    query_vec = vectorize(normalized_query, dictionary)
    scores = []
    for doc_info, doc_vec in doc_term_matrix.items():
        sim = similarity(query_vec, doc_vec)
        scores.append((sim, doc_info))
    scores.sort(reverse=True)

    return scores
```

Trong đó:

- **Dòng 1:** Khai báo hàm **ranking()** nhận tham số đầu vào là câu truy vấn, bộ từ vựng và ma trận doc term.
- **Dòng 2, 3:** Thực hiện vector hóa văn bản.
- **Dòng 4:** Khởi tạo danh sách dùng để chứa kết quả cosine similarity.
- **Dòng 5, 6, 7:** Duyệt qua từng tài liệu trong ma trận doc-term, thực hiện tính cosine similarity giữa vector câu truy vấn và vector tài liệu đang xét. Sau đó, đưa kết quả này vào danh sách **scores**.
- **Dòng 8:** Sắp xếp kết quả cosine similarity theo thứ tự giảm dần. Từ đây ta tìm được ranked list.
- **Dòng 10:** Trả về ranked list.

9. **Thực hiện truy vấn bất kì:** Với tất cả các thành phần đã xây dựng ở trên, ta có thể triển khai một đoạn code truy vấn văn bản với nội dung bất kì như sau:

```
query_lst = ['what is the official language in Fiji']
top_k = 10
for query in query_lst:
    scores = ranking(query, dictionary, doc_term_matrix)
    print(f'Query: {query}')
    print('=== Relevant docs ===')
    for idx in range(top_k):
        doc_score = scores[idx][0]
        doc_content = scores[idx][1][0]

        print(f'Top {idx + 1}; Score: {doc_score:.4f}')
        print(doc_content)
        print('\n')
```

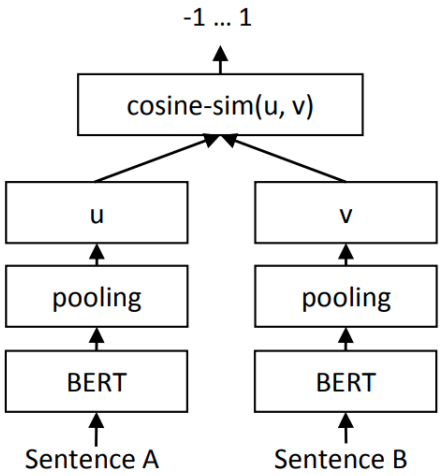
Kết quả mà đoạn code này trả về như sau:

```
Query: what is the official language in Fiji
=== Relevant docs ===
Top 1; Score: 0.6556
The official languages in Fiji are Fijian and English. A dialect of Hindustani is also widely spoken among Indo-Fijians.


Top 2; Score: 0.6556
The official languages in Fiji are Fijian and English. A dialect of Hindustani is also widely spoken among Indo-Fijians.


Top 3; Score: 0.5715
The official languages. Fiji's 1997 Constitution established Fijian as one of the official languages of the country. Fij


Top 4; Score: 0.5604
Of all the languages of Russia, Russian is the only official language. There are 35 different languages which are consid


Top 5; Score: 0.5592
Finnish is one of two official languages of Finland (the other being Swedish, spoken by 5.42% of the population as of 20


Top 6; Score: 0.5307
Liberia is a multilingual country where more than thirty languages are spoken. English is the official language. None of


Top 7; Score: 0.5164
There are over 120 ethnic groups, each with its own language or dialect. Indigenous Tanzanians make up 99 % of the popul


Top 8; Score: 0.5094
the two official languages every country in the world has official languages that are spoken amongst the population thes


Top 9; Score: 0.4811
Fiji Language. The Fijian language spoken in Fiji is a type of Austronesian langauge and is part of the Malayo-Polynesia


Top 10; Score: 0.4547
Upon Uganda's independence in 1962, English was maintained as the official language, as it was already rooted deeply in
```

Hình 10: Top 10 văn bản có độ tương đồng cao nhất với câu truy vấn "what is the official language in Fiji".

Optional: Semantic Search với pretrained BERT

Mô hình Vector Space Model dựa trên các đặc trưng thống kê để xây dựng một dạng biểu diễn phù hợp cho các văn bản, từ đó thực hiện xếp hạng truy vấn dựa trên giá trị tương đồng giữa vector câu truy vấn và vector tài liệu. Cách tiếp cận này song lại không thể nắm bắt được các thông tin về mặt ngữ nghĩa trong một câu truy vấn, dẫn đến không có một kết quả tìm kiếm tốt hơn khi câu truy vấn trở nên phức tạp. Chính vì vậy, một đề xuất sử dụng Encoder của mô hình transformer để tạo vector biểu diễn ra đời. Với việc tận dụng sức mạnh của mô hình, ta có cải thiện đáng kể kết quả truy vấn của mô hình vector space model hiện tại.



Hình 11: Ý tưởng sử dụng pretrained BERT để thực hiện truy vấn. Phần này sẽ được đề cập trong buổi project. Nguồn: link

.

Phần III: Trắc nghiệm

1. Input của một hệ thống Text Retrieval bao gồm?

   (*a*) Câu truy vấn.
   (*b*) Câu truy vấn và bộ ngữ liệu.
   (*c*) Câu truy vấn và bộ từ vựng.
   (*d*) Bộ ngữ liệu.

2. Module nào sau đây không thuộc trong pipeline chung của một hệ thống Text Retrieval?

   (*a*) Create doc-term matrix
   (*b*) Indexing
   (*c*) Text Normalization
   (*d*) Ranking

3. Bộ ngữ liệu là gì?

   (*a*) Danh sách các câu truy vấn.
   (*b*) Danh sách các từ vựng.
   (*c*) Danh sách các tài liệu.
   (*d*) Danh sách các stopwords.

4. Kỹ thuật nào sau đây không phải là một kỹ thuật chuẩn hóa văn bản?

   (*a*) Lowercasing
   (*b*) Uppercasing
   (*c*) Stopwords Removal
   (*d*) Vectorization

5. Các bước nào sau đây không nằm trong các bước xây dựng bộ từ vựng?

   (*a*) Tokenization
   (*b*) Stemming
   (*c*) Text Normalization
   (*d*) Get unique words

6. Trong kỹ thuật bag-of-words, các giá trị của một vector biểu diễn có ý nghĩa gì so với văn bản gốc?

   (*a*) Từ vựng có tồn tại trong văn bản hay không
   (*b*) Từ vựng có nghĩa hay không
   (*c*) Số lần xuất hiện của từ vựng trong văn bản
   (*d*) Số lần xuất hiện của từ vựng trong bộ từ vựng

7. Bước nào sau đây không nằm trong các bước xây dựng bộ từ vựng?

   (*a*) Tokenization
   (*b*) Stemming
   (*c*) Text Normalization
   (*d*) Get unique words

8. Cho danh sách các tài liệu với vector biểu diễn như sau:

   - doc_1: $[2, 1, 0, 0, 3, 2]$
   - doc_2: $[2, 0, 1, 1, 0, 0]$
   - doc_3: $[1, 1, 1, 1, 1, 1]$
   - doc_4: $[1, 2, 3, 0, 0, 0]$

   Với câu truy vấn có vector biểu diễn là $[3, 1, 0, 0, 2, 1]$, tài liệu có độ tương đồng cao thứ 2 so với câu truy vấn theo độ đo cosine similarity là?

(*a*) doc_1
(*b*) doc_2
(*c*) doc_3
(*d*) doc_4

- Hết -