P05 Project Documentation

(By Swadha Arora & Kusum Meghrajani)

Use Case 2: Query expansion with WordNet

Indexing plays a critical role in the functioning of any search engine, as efficient indexing enhances information retrieval. In our search engine, we utilize Ranked Retrieval with the Elasticsearch framework, which is built on the Lucene library.

To begin, we launch the Elasticsearch server locally and instantiate an Elasticsearch object to access its features.

Our dataset consists of 1,000 text files, each representing information about several different types of city names such as ‘Paris’, ‘New York’, ‘Turkey’ etc. These documents are sourced from Wikipedia which are indexed to facilitate retrieval.

We create an inverted index by importing these text files as documents. When a user inputs a query, each term in the query is assigned attributes such as boost and weight. These attributes are crucial for calculating the document scores. The score of each document hit is determined by multiplying the term's weight by the document's relevance. Additionally, we can explicitly define how the scores are calculated, allowing us to adjust term weights to influence document rankings. This ensures that the most relevant documents appear at the top, followed by others in descending order of relevance.

The search results are displayed based on the user's query. However, users may sometimes require information on topics closely related to their original query. To address this, we implement query expansion. Using WordNet, a lexical database, we identify synonyms for the original query and append them to it. The revised query, consisting of the original terms and user-selected synonyms, is then executed, providing a broader range of relevant documents.

Since we want the original query results at the top and the results using the expanded query after it, we do that by modifying the weights of the original
and the appended terms, where the original query terms will have a higher weight than the other terms.

To run the program, you would need the following components:

Backend:
Elasticsearch
NLTK (Wordnet to be specific) using:
~ import nltk
~ nltk.download(‘wordnet’)
~ nltk.download('omw-1.4')
Django

Frontend:
HTML, CSS, JavaScript
Bootstrap

Once this is done, you would also need the dataset which has been obtained by using a web crawler and collecting data from Wikipedia for the list of 1000 cities.

For using Elasticsearch, it needs to be downloaded and running on the localhost which can be done the command: **path\bin\elasticsearch.bat**

You run the python file which consists of all the code and uses function **‘query_results()’** where you pass the query as a parameter, along with optional parameters such as weights for original query, expanded query and total documents to be displayed at once.

The result would be a dictionary that consists of two elements: suggest and docs. Suggest has a list of words that it would suggest for the user and docs has a list of all the hits within the threshold ordered by rank.

Code Documentation

(By Swadha Arora & Kusum Meghrajani)

For Elasticsearch:

1. We import Elasticsearch and WordNet along with the os library which we require for data manipulation.

The '*syn()*' function receives the query word as a parameter once the user submits its query and using the '*wordnet.sysnets()*' function, it returns a list of synonyms to the query word which is used for query expansion.

```
synsets = wordnet.synsets(word)
if not synsets:
    return {"synonyms": [], "definitions": [], "related_words": [], "message": "No data found for this word in Word


for syn in synsets:
    for lemma in syn.lemmas():
        synonyms.add(lemma.name())
        if lemma.antonyms():
            synonyms.add(lemma.antonyms()[0].name())
```

2. The core of the entire program is the '*query_results*' function which takes the *'query_input'* variable from the front end i.e. user query. It receives the query from the user.

We search the query throughout the documents by using the search() function which returns us hits by the query and displays the result.

```python
def query_results(request):
    query = request.GET.get("q", "")

    if not query:
        return render(
            request,
            "search/query_results.html",
            {"query": query, "results": [], "nltk_info": None, "random_diseases": []},
        )

    results = []


    try:
        response = es.search(
            index="disease_data",
            body={"query": {"match": {"content": query}}},
        )
    except Exception as e:
        return render(
            request,
            "search/query_results.html",
            {
                "query": query,
                "results": [{"title": "Error", "content": str(e)}],
                "nltk_info": None,
                "random_diseases": [],
            },
        )
```

3. To perform query expansion, we go through the entire query by splitting the query term (eg. ‘New York’ would be ‘New’, ‘York’) and look up synonyms for each term.

```python
def query_results(request):
    results = []
    expanded_query = []  # To collect original terms and their synonyms
    terms = query.split()  # Split the query into individual words

    try:
        # Expand query terms with synonyms
        for term in terms:
            synonyms_data = get_wordnet_info(term)  # Fetch WordNet info for the term

            synonyms = synonyms_data.get("synonyms")
            expanded_query.append(term)  # Add the original term
            expanded_query.extend(syn for syn in synonyms if syn not in expanded_query)  # Add unique synonyms

        # Construct the Elasticsearch query with boosted weights
        weight_original = 100
        weight_synonyms = 70

        should_clauses = [
            {"match": {"content": {"query": term, "boost": weight_original}}} for term in terms
        ]
        should_clauses += [
            {"match": {"content": {"query": synonym, "boost": weight_synonyms}}} for synonym in expanded_query if synon
        ]

        # DEBUG: Print the constructed Elasticsearch query body
        body = {"query": {"bool": {"should": should_clauses}}}

        # Perform the search in Elasticsearch
        response = es.search(index="city_data", body=body, size=10)
```

For Django:

Django, written in Python, is an open source framework used for development of websites. We have used Django to connect the website with the backend code for the information retrieval. Once the query is written and the “Search” button is pressed, the data is sent to the backend. With the help of Django, this data is used with the backend program and the output data is implemented back on the website.

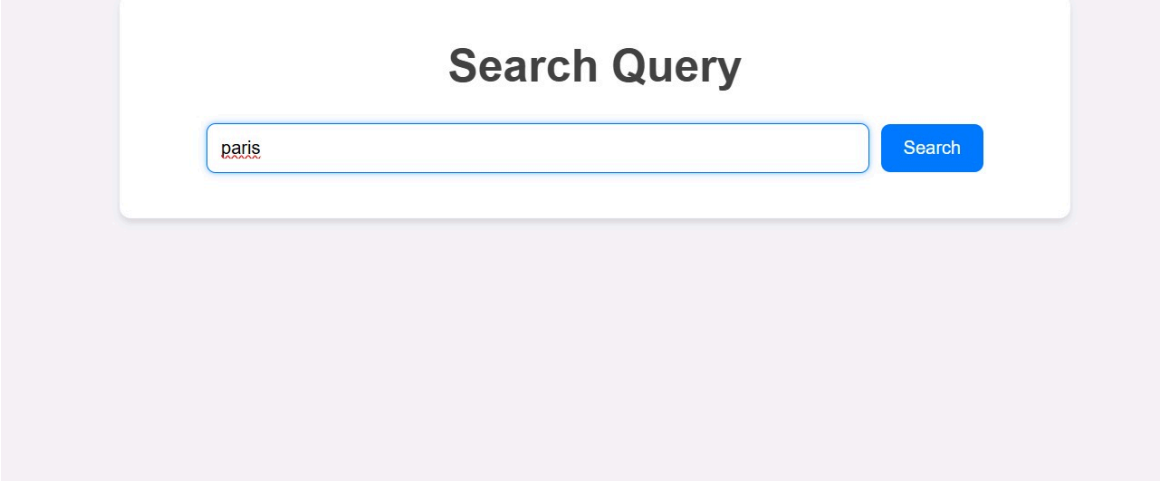
Instructions on running the code:

1. Install and run elasticsearch 7.16.3 on your local system along with Django (latest version).
2. Create a virtual environment: “python -m venv venv” in root directory
3. Activate the virtual environment: ”venv\Scripts\activate”
4. Install the required Python packages: “pip install -r requirements.txt”
5. Index data in Elasticsearch: “python index_data.py”
6. Run migrations to create the database: “python manage.py migrate”
7. Populate the database with disease names: “python manage.py import_diseases”
8. Go to the Django directory (root directory) and run ‘python manage.py runserver’ (Please make sure that the dependencies are installed and downloaded)
9. Go to http://127.0.0.1:8000/ where you shall get the hosted website/homepage of the search engine.
10. Each term can be searched with results coming below with highlighted words of query as well as suggested synonyms for query expansion.

“Okay, but how do I use it?”

When you visit the homepage of the search engine, you shall see a search bar. Input your query in there and simply click on search. Once the search button is clicked, the engine shall fetch you related documents.

Also just below the search bar, you shall see Synonyms, related data, and related terms to your query.



Results for "paris"

Also Search for : Hubli , Tagum , Victorville

paris

Paris (French pronunciation: [paʁi] (listen)) is the capital and most populous city of France, with an estimated population of 2,175,601 residents as of 2018, in an area of more than 105 square kilometres (41 square miles). Since the 17th century, **Paris** has been one of Europe's major centres of finance, diplomacy, commerce, fashion, gastronomy, science, and arts. The City of **Paris** is the centre and seat of government of the region and province of Île-de-France, or **Paris** Region, which has an estimated population of 12,174,880, or about 18 percent of the population of France as of 2017. The **Paris** Region had a GDP of €709 billion ($808 billion) in 2017. According to the Economist Intelligence Unit Worldwide Cost of Living Survey in 2018, **Paris** was the second most expensive city in the world, after Singapore and ahead of Zürich, Hong Kong, Oslo, and Geneva. Another source ranked **Paris** as most expensive, on par with Singapore and Hong Kong, in 2018.**Paris** is a major railway, highway, and air-transport hub served by two international airports: **Paris**–Charles de Gaulle (the second-busiest airport in Europe) and **Paris**–Orly. Opened in 1900, the city's subway system, the **Paris** Métro, serves 5.23 million passengers daily; it is the second-busiest metro system in Europe after the Moscow Metro. Gare du Nord is the 24th-busiest railway station in the world, but the busiest located outside Japan, with 262 million passengers in 2015. **Paris** is especially known for its museums and architectural landmarks: the Louvre received 2.8 million visitors in 2021, despite the long museum closings caused by the COVID-19 virus. The Musée d'Orsay, Musée Marmottan Monet and Musée de l'Orangerie are noted for their collections of French Impressionist art. The Pompidou Centre Musée National d'Art Moderne has the largest collection of modern and contemporary art in Europe. The Musée Rodin and Musée Picasso exhibit the works of two noted **Paris**ians. The historical district along the Seine in the city centre has been classified as a UNESCO World Heritage Site since 1991; popular landmarks there include the Cathedral of Notre Dame de **Paris** on the Île de la Cité, now closed for renovation after the 15 April 2019 fire. Other popular tourist sites include the Gothic royal chapel of Sainte-Chapelle, also on the Île de la Cité; the Eiffel Tower, constructed for the **Paris** Universal Exposition of 1889; the Grand Palais and Petit Palais, built for the **Paris** Universal Exposition of 1900; the Arc de Triomphe on the Champs-Élysées, and the hill of Montmartre with its artistic history and its Basilica of Sacré-Coeur **Paris** received 12.6 million visitors in 2020, measured

Additional Information:

Synonyms:

French_capital

capital_of_France

Paris

genus_Paris

City_of_Light

Definitions:

(Greek mythology) the prince of Troy who abducted Helen from her husband Menelaus and provoked the Trojan War

the capital and largest city of France; and international center of culture and commerce

sometimes placed in subfamily Trilliaceae

a town in northeastern Texas