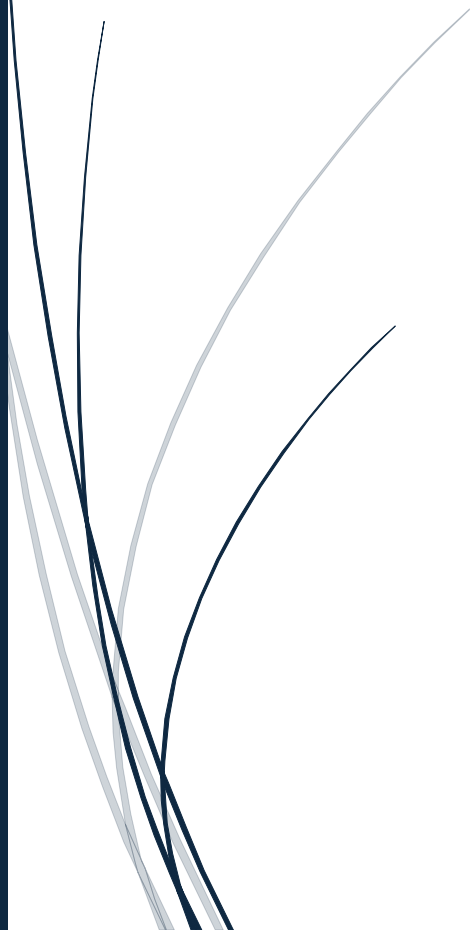


07/01/2025

Rapport de projet

Développement d'un moteur de
recherche en Python



Timothy MARCIA, Elias AIT HASSOU
Master 1 Informatique - Université Lumière Lyon II

Table des matières

| | |
|-----------------------------------|---|
| Présentation du projet | 2 |
| Partie conceptuelle | 3 |
| L'environnement de travail | 3 |
| Diagramme des classes | 4 |
| Mise en place du projet | 5 |
| Répartition des tâches | 5 |
| Les difficultés rencontrées | 6 |
| Exemple d'utilisation | 7 |
| Validation du logiciel | 8 |
| Tests unitaires | 8 |
| Conclusion et perspectives | 9 |

Présentation du projet

Lien vers le GitHub : <https://github.com/aithassouelias/moteur-recherche-python>

Ce projet a pour objectif de développer un moteur de recherche textuel sur des données touristiques en utilisant le langage de programmation Python. Le moteur de recherche permet aux utilisateurs de rechercher des activités qu'ils souhaitent réaliser et retourne des descriptions de villes où ces activités sont disponibles. Ce logiciel repose sur la collecte, le traitement, le stockage, l'analyse et la restitution des données.

Pour constituer le corpus de données, nous avons exploité l'API WikiVoyage. Cette API fournit des descriptions des activités touristiques proposées dans différentes villes à travers le monde. Ces données textuelles représentent la base de notre moteur de recherche.

L'objectif principal est de proposer une interface permettant :

- À l'utilisateur de saisir une activité spécifique qu'il aimerait réaliser.
- D'analyser les descriptions des villes contenues dans le corpus
- De restituer les résultats sous forme d'une liste des villes pertinentes accompagnées de leurs descriptions.

Partie conceptuelle

L'environnement de travail

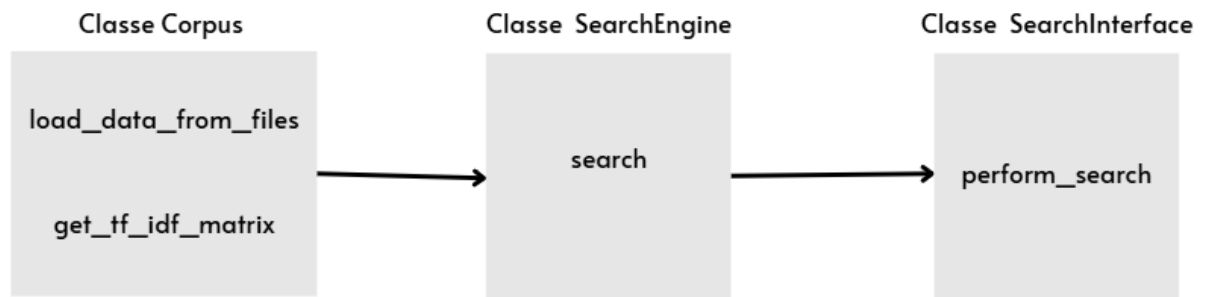
Pour le développement de notre moteur de recherche, nous avons choisi un environnement de travail adapté aux exigences de collecte, traitement et d'analyse des données textuelles.

Le projet a été implémenté en Python, étant le langage de programmation étudié lors des séances de cours. Sa simplicité, et ses bibliothèques dédiées au traitement des données textuelles ont été essentielles pour mener à bien ce projet.

Les différentes bibliothèques utilisées ont des rôles spécifiques décrit ci-après :

- **Requests** pour interagir avec l'API WikiVoyage
- **Pandas** pour la création de DataFrame
- **JSON** pour lire et écrire des fichiers JSON tels que les réponses d'API ou nos fichiers de stockage des données.
- **NLTK** pour récupérer une liste des termes à faible valeur ajoutée dans la langue anglaise nous permettant de nettoyer le corpus.
- **NumPy** pour transformer les requêtes textuelles et les documents en vecteurs.
- **Regex** pour mettre en place des expressions régulières nous permettant de nettoyer nos données textuelles en retirant les nombres et les caractères spéciaux.
- **Tkinter** pour réaliser l'interface graphique finale permettant d'effectuer une recherche et afficher les résultats.

Diagramme des classes



Nous avons 3 classes qui interagissent entre elles pour mener à bien la recherche de documents. La première classe corpus permet de récupérer le corpus depuis les fichiers JSON, une méthode permet d'avoir des statistiques sur les données à disposition. L'objectif principale de cette classe est de construire la matrice TF-IDF.

La classe SearchEngine permet de réaliser une recherche en calculant les coefficients de similarités à partir d'un objet corpus et de sa matrice TF_IDF. Enfin, la classe SearchInterface permet à un utilisateur de réaliser une recherche depuis une interface graphique Tkinter à l'aide de la classe SearchEngine.

Mise en place du projet

Répartition des tâches

Pour mener à bien ce projet, nous avons travaillé en parfaite collaboration tout au long de celui-ci. La recherche d'idée de données que l'on voulait traiter ainsi que la manière dans le logiciel allait être réalisé a été discuté ensemble lors des dernières séances de TD. L'un des deux a débuté par développer les fonctions de collecte de données pendant que l'autre anticipait les questions de nettoyage des données.

Par la suite, nous nous sommes réparti le développement des différentes classes et des différentes méthodes de manière équitable pour que chacun des deux puisse toucher aux différentes parties du projet et avoir une compréhension globale de ce qui a été produit.

Les difficultés rencontrées

L'une des difficultés rencontrées lors de ce projet a été la qualité des données textuelles. En effet, nous avons collecté les données et les avons stockées dans un fichier JSON, nous avons implémenté une méthode permettant d'afficher des statistiques sur le corpus. Celle-ci nous a permis d'avoir un aperçu des données que nous avons à notre disposition.

Ayant récolté des données sur des activités relatives à des villes dans le monde entier, certaines descriptions comportaient des caractères provenant d'autres alphabets et qui étaient donc inutile dans le cadre de notre projet. Afin de retirer toutes ces données inutiles, nous avons mis en place des expressions régulières afin de conserver uniquement les caractères de notre alphabet. Cette solution nous a permis également d'éviter d'allonger le temps de traitement des méthodes d'analyse du corpus.

Une fois que nous avons retiré ces caractères, nous avons décidé de mettre en place une méthode nous permettant de visualiser quelles étaient les dix termes les plus utilisés dans le corpus.

```
=====
Statistiques du Corpus
=====
Nombre de villes contenues dans le corpus : 92
Nombre moyen de caractères à traiter par ville : 4026

Les 10 mots les plus fréquents dans le corpus :
the: 4319
and: 2045
in: 1758
of: 1757
a: 1441
to: 1275
is: 1045
are: 752
for: 591
at: 587
```

```
=====
Statistiques du Corpus
=====
Nombre de villes contenues dans le corpus : 92
Nombre moyen de caractères à traiter par ville : 2853

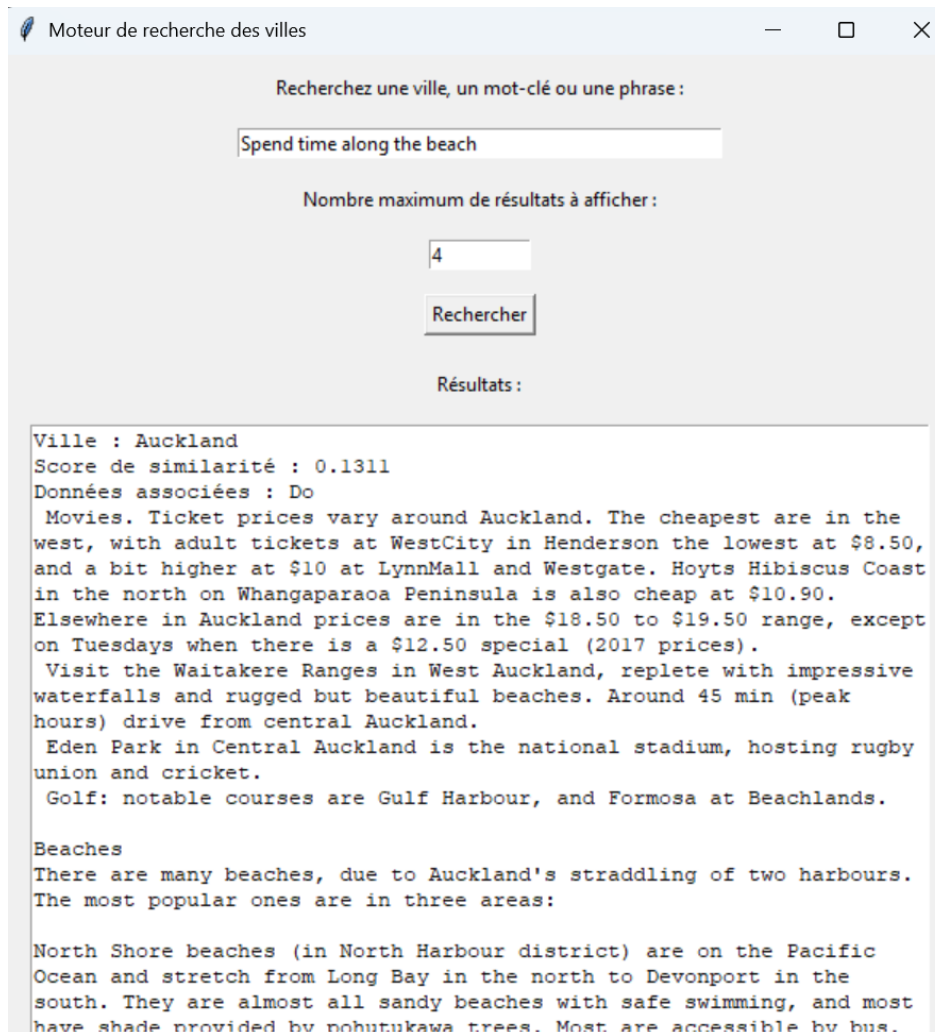
Les 10 mots les plus fréquents dans le corpus :
city: 390
one: 204
park: 203
play: 149
music: 141
beach: 140
stadium: 128
see: 123
day: 122
football: 117
```

Comme nous pouvons l'observer, avant le nettoyage des données, les dix termes les plus utilisés étaient des mots de liaison. Ces derniers n'étant pas pertinents dans le cadre de notre analyse, nous avons donc décidé de les retirer en utilisant la liste de 'stop words' mise en place par la bibliothèque NLTK et en l'enrichissant avec d'autres termes spécifiques à notre corpus.

Exemple d'utilisation

Afin de lancer l'interface graphique permettant d'effectuer une recherche, il faut exécuter le fichier `main.py` disponible à la racine du projet.

Après quelques secondes, cette interface graphique s'affichera. Vous pourrez alors y entrer les mots clés à rechercher ainsi que le nombre de résultats que vous souhaitez afficher, enfin il faudra cliquer sur le bouton « Rechercher » pour lancer la recherche de documents.



Recherchez une ville, un mot-clé ou une phrase :

Spend time along the beach

Nombre maximum de résultats à afficher :

4

Rechercher

Résultats :

Ville : Auckland
Score de similarité : 0.1311
Données associées : Do
Movies. Ticket prices vary around Auckland. The cheapest are in the west, with adult tickets at WestCity in Henderson the lowest at \$8.50, and a bit higher at \$10 at LynnMall and Westgate. Hoyts Hibiscus Coast in the north on Whangaparaoa Peninsula is also cheap at \$10.90. Elsewhere in Auckland prices are in the \$18.50 to \$19.50 range, except on Tuesdays when there is a \$12.50 special (2017 prices).
Visit the Waitakere Ranges in West Auckland, replete with impressive waterfalls and rugged but beautiful beaches. Around 45 min (peak hours) drive from central Auckland.
Eden Park in Central Auckland is the national stadium, hosting rugby union and cricket.
Golf: notable courses are Gulf Harbour, and Formosa at Beachlands.

Beaches
There are many beaches, due to Auckland's straddling of two harbours. The most popular ones are in three areas:

North Shore beaches (in North Harbour district) are on the Pacific Ocean and stretch from Long Bay in the north to Devonport in the south. They are almost all sandy beaches with safe swimming, and most have shade provided by pohutukawa trees. Most are accessible by bus.

Afin de voir l'ensemble des résultats obtenues, veuillez cliquer dans la zone résultat et descendre afin de voir les différents documents retournés.

Il est important d'effectuer ses recherches en utilisant des mots clés relatifs aux activités touristiques et en anglais.

Validation du logiciel

Tests unitaires

Exemple d'un test unitaire de notre fonction `get_city_activities` qui a échoué dans notre classe `Corpus` :

```
=====
ERROR: test_get_city_activities (_main_.TestCorpus.test_get_city_activities)
Test pour la méthode get_city_activities.
=====
Traceback (most recent call last):
  File "c:\Users\timot\OneDrive\Desktop\Travail\Code\specialitePython\projet\moteur-recherche-python\models\test_corpus.py.py", line 35, in test_get_city_activities
    activities_not_found = self.corpus.get_city_activities("Berlin")
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\timot\OneDrive\Desktop\Travail\Code\specialitePython\projet\moteur-recherche-python\models\Corpus.py", line 39, in get_city_activities
    activities = city.info.get("do", "City non")
    ^^^^^^^^^^^
AttributeError: 'str' object has no attribute 'get'
=====
```

Nous pouvons voir que sur la globalité des méthodes de la classe `Corpus`, nous avons :

```
FAILED (failures=1, errors=2)
```

Après correction des méthodes nous arrivons à faire passer des tests unitaires :

```
FAILED (failures=1, errors=1)
```

Malheureusement nous n'avons pas réussi à régler le problème sur toutes les méthodes de notre application.

Conclusion et perspectives

Le développement de ce moteur de recherche textuel nous a permis d'appliquer les connaissances que nous avons vu en cours. Malgré les difficultés rencontrées, notamment liés à la qualité des données textuelles et aux difficultés d'implémentation de certaines fonctionnalités, nous avons réussi à concevoir une application fonctionnelle qui répond aux objectifs.

Ce projet nous a permis d'améliorer nos compétences en programmation, en gestion de données non structurées et en validation logicielle à travers des tests unitaires. Bien que certaines méthodes restent perfectibles, les solutions mises en œuvre, comme le nettoyage des données, ont apporté une plus grande pertinence aux résultats.

Perspectives :

1. **Extension linguistique** : Étendre le moteur pour supporter des recherches en plusieurs langues, en s'appuyant sur des outils de traduction et des corpus multilingues.
2. **Amélioration des méthodes** : Retravailler les méthodes afin que les tests unitaires ne renvoient plus d'erreur.
3. **Déploiement** : Héberger l'application sur une plateforme en ligne pour permettre à des utilisateurs externes de l'explorer.

En conclusion, ce projet nous a offert une expérience enrichissante et nous a permis de relever des défis techniques variés. Il constitue une base solide pour des développements futurs dans le domaine des moteurs de recherche et du traitement du langage naturel.