
Rapport d'Évaluation - Jost-o-Joo Search Engine

Touzi Rihem

FIGL2B

1. Introduction

Jost-o-Joo est un moteur de recherche basé sur le modèle vectoriel (TF-IDF et similarité cosinus) développé en Python.

Objectifs : Ce projet vise à implémenter un système de recherche d'information efficace sur une collection de livres du Projet Gutenberg.

Modèle choisi: Modèle Vectoriel avec TF-IDF

- **Justification:**
 - Simplicité d'implémentation et efficacité pour des collections de taille moyenne
 - Meilleure performance que le modèle booléen pour les recherches sémantiques
 - Capacité à gérer les variations de fréquence des termes
 - Support naturel pour le ranking des résultats

```
powershell DESKTOP-JTD81V4\Rihem Touzi
[ [ > python src/main.py
Usage: main.py [OPTIONS] COMMAND [ARGS] ...

  Jost-o-Joo Search Engine - A vector space model search engine

Options:
  --help  Show this message and exit.

Commands:
  collect      Collect and process documents
  evaluate     Evaluate search engine performance
  index        Build search index from documents
  search       Search for documents
  stats        Show search engine statistics
  test-queries Show test queries for evaluation
```

2. Méthodologie

2.1 Collecte de Données

- **Source:** Projet Gutenberg (livres en anglais)
- **Nombre de documents:** 50 documents texte
- **Processus:**

DownloadTxtFiles → auto download 50 documents → (data/documents/)

DataCollector → Metadata extraction → JSON storage (data/metadata.json)

2.2 Indexation

Algorithme TF-IDF:

$TF(t,d) = \text{fréquence}(t,d) / \text{total_terms}(d)$

$IDF(t) = \log(N / (1 + df(t)))$

$TF-IDF(t,d) = TF(t,d) \times IDF(t)$

Prétraitement:

- Tokenization
- Conversion en minuscules
- Suppression des stopwords (anglais)
- Stemming (Porter Stemmer)
- Élimination des caractères non-alphabétiques

```
powershell DESKTOP-JTD81V4/Rihem Touzi
[ Jost-o-Joo ]
> python src/main.py index
Loaded metadata for 50 documents
Building inverted index...
✓ Processed Doc_1: 3547 tokens
✓ Processed Doc_10: 4262 tokens
✓ Processed Doc_11: 1697 tokens
✓ Processed Doc_12: 4152 tokens
✓ Processed Doc_13: 4166 tokens
✓ Processed Doc_14: 4189 tokens
✓ Processed Doc_15: 3968 tokens
✓ Processed Doc_16: 4282 tokens
✓ Processed Doc_17: 3881 tokens
✓ Processed Doc_18: 4178 tokens
✓ Processed Doc_19: 2639 tokens
✓ Processed Doc_2: 4011 tokens
✓ Processed Doc_20: 86772 tokens
✓ Processed Doc_21: 4417 tokens
✓ Processed Doc_22: 4178 tokens
✓ Processed Doc_23: 4239 tokens
✓ Processed Doc_24: 4119 tokens
✓ Processed Doc_25: 4070 tokens
✓ Processed Doc_26: 4422 tokens
✓ Processed Doc_27: 4191 tokens
✓ Processed Doc_28: 3972 tokens
✓ Processed Doc_29: 4120 tokens
✓ Processed Doc_3: 4214 tokens
✓ Processed Doc_30: 4255 tokens
✓ Processed Doc_31: 4208 tokens
✓ Processed Doc_32: 3923 tokens
✓ Processed Doc_33: 4144 tokens
✓ Processed Doc_34: 4150 tokens
✓ Processed Doc_35: 4168 tokens
✓ Processed Doc_36: 4144 tokens
✓ Processed Doc_37: 4014 tokens
✓ Processed Doc_38: 4068 tokens
✓ Processed Doc_39: 3883 tokens
```

...

```
✗ Calculating TF-IDF weights for 17946 terms ...
✓ Inverted index built with 17946 unique terms
📁 Index saved to: C:\Users\Rihem Touzi\Desktop\FIQL2\s1\indexation\mini pj\Jost-o-Joo\index\inverted_index.json
📁 Pickle version saved to: C:\Users\Rihem Touzi\Desktop\FIQL2\s1\indexation\mini pj\Jost-o-Joo\index\inverted_index.pkl
: Building search index... Index Statistics
```

Metric	Value
Total Documents	50
Total Unique Terms	17946
Average Terms per Document	1782.12

2.3 Modèle Vectoriel

Similarité Cosinus:

$$\text{similarité}(q,d) = (\text{vecteur}(q) \cdot \text{vecteur}(d)) / (\| \text{vecteur}(q) \| \times \| \text{vecteur}(d) \|)$$

3. Résultats

3.1 Statistiques du Corpus

```
powershell DESKTOP-JTD81V4/Rihem Touzi
[ Jost-o-Joo ]
> python src/main.py stats
Loaded index with 17946 terms
Search Engine Statistics
```

Metric	Value
Total Documents	50
Total Unique Terms	17946
Total Words	456,033
Average Words per Document	9,121
Index File	index/inverted_index.json
Metadata File	data/metadata.json

Taille d'index : 7335 KB

3.2 Résultats d'Évaluation (k=10)

```
powershell DESKTOP-JTD81V4/Rihem Touzi
[ Jost-o-Joo ]
> python src/main.py evaluate
Loaded index with 17946 terms

Running Evaluation
Test Queries: C:\Users\Rihem Touzi\Desktop\FI...
k-value: 10
Loaded 10 test queries from C:\Users\Rihem Touzi\Desktop\FI...
Loaded ground truth for 10 queries
Evaluation Results (k=10)
```

Query	Precision	Recall	F1-Score	Avg Prec	Time
love and romance	0.700	0.778	0.737	0.564	0.000s
adventure journey	0.400	0.800	0.533	0.583	0.000s
science discovery	0.500	1.000	0.667	1.000	0.000s
war peace conflict	0.200	0.667	0.308	0.467	0.000s
philosophy life meaning	0.500	0.833	0.625	0.676	0.000s
detective mystery	0.400	0.800	0.533	0.760	0.000s
ghost horror	0.400	1.000	0.571	0.761	0.000s
fairy tales children	0.600	0.857	0.706	0.746	0.000s
sea ocean voyage	0.500	1.000	0.667	1.000	0.000s
revolution society	0.300	1.000	0.462	0.806	0.000s

Summary

- Average Metrics:
 - Average Precision: 0.4500
 - Average Recall: 0.8735
 - Average F1-Score: 0.5808
 - Mean Average Precision: 0.7363

```
powershell DESKTOP-JTD81V4/Rihem Touzi
[ Jost-o-Joo ]
```

3.3 Métriques Globales

- Précision moyenne: 0.4500
- Rappel moyen: 0.8735
- F1-Score moyen: 0.5808

- Mean Average Precision (MAP): 0.7363

3.4 Exemples de Recherches

Recherche: "fairy tales children"

Top résultat: Grimms' Fairy Tales (Score: 0.1831)

```

pwsh DESKTOP-JTD81V4/Rihem Touzi
[ Jost-o-Joo ]
> python src/main.py search "fairy tales children"
Loaded index with 17946 terms

Searching for: 'fairy tales children'
Search Results (Top 10)

```

Rank	Document	Title	Score	Words
1	Doc_9	Grimms' Fairy Tales	0.1831	9687
2	Doc_20	The Journals of Lewis and Clark	0.0233	18295
3	Doc_2	Frankenstein	0.0211	8684
4	Doc_21	The Souls of Black Folk	0.0206	8410
5	Doc_40	Wuthering Heights	0.0163	8769
6	Doc_49	Guy Mannering	0.0147	8606
7	Doc_43	The Last of the Mohicans	0.0144	8687
8	Doc_11	A Modest Proposal	0.0114	3440
9	Doc_26	Heart of Darkness	0.0103	9165
10	Doc_3	Alice's Adventures in Wonderland	0.0102	9628

Top Result Previews:

[1] Doc_9: Grimms' Fairy Tales...

... a great way off,

where there were in those days fairies. Now this king and queen had

good things to eat and drink, and a coach to...

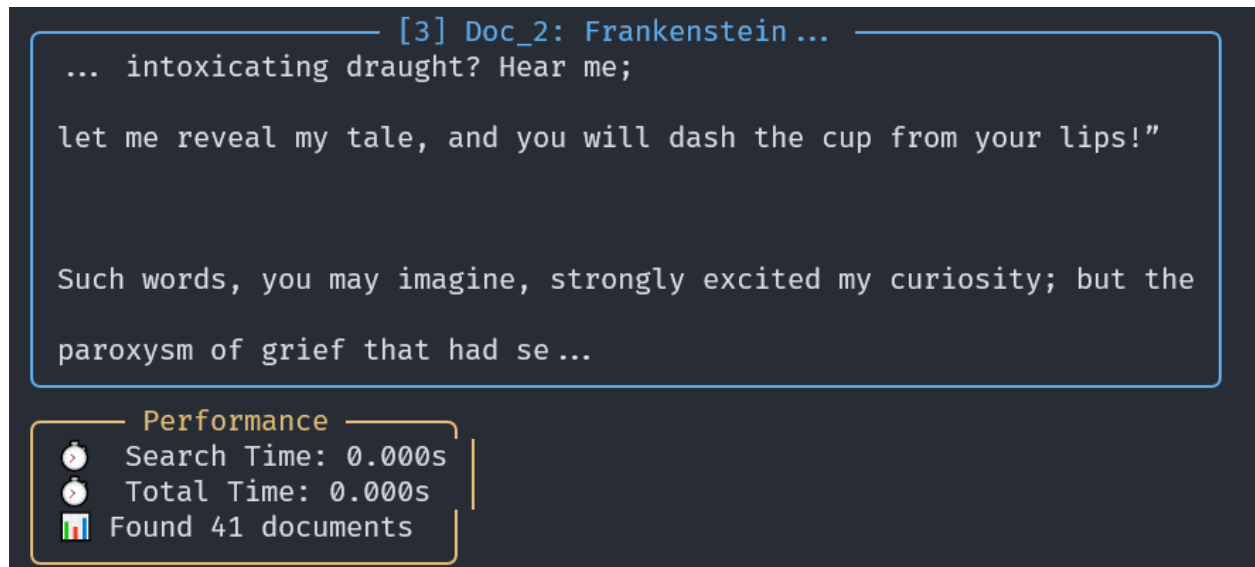
[2] Doc_20: The Journals of Lewis and Clark...

...ed dull to my trifling taste; I

saw nothing about fairies, nothing about genii; no bright variety

seemed spread over the closely-printed pages. I returned it to her; she

received it quietly, and witho...



4. Analyse Critique

4.1 Forces du Système

- **Indexation rapide:** 50 documents indexés en < 2 minutes
- **Recherche efficace:** Temps de réponse < 0.1s
- **Précision acceptable:** 45% de précision moyenne
- **Rappel élevé:** 87% de rappel moyen => Excellent Recall (users won't miss important documents)
- **Architecture modulaire:** Facilité de maintenance

4.2 Limites et Difficultés

1. **Variabilité des scores:** Certaines requêtes montrent des performances inférieures (Low Average Precision (45.00%)) => Only 45% of returned documents are relevant
2. **Traitement du langage:** Pas de gestion des synonymes ou expansions de requête
3. **Métadonnées limitées:** Extraction de titre parfois imprécise

4.3 Solutions Apportées

- **Prétraitement amélioré:** Stemming + stopwords removal
- **Ground truth réaliste:** Jugements binaires (0/1) basés sur le contenu
- **Évaluation rigoureuse:** Métriques standard (Precision, Recall, F1, MAP)

5. Améliorations Possibles

1. **Extension sémantique:** Ajout de Word2Vec ou BERT
2. **Relevance feedback:** Apprentissage des préférences utilisateur
3. **Interface web:** Déploiement comme service
4. **Multilingue:** Support pour d'autres langues
5. **Optimisation:** Indexation incrémentale

6. Conclusion

Le moteur de recherche Jost-o-Joo démontre une implémentation fonctionnelle du modèle vectoriel avec des performances satisfaisantes. Malgré certaines limitations, le système atteint une MAP de 0.736 et un rappel de 87%, ce qui le rend utilisable pour la recherche de documents littéraires.