M2 DSC - Université Jean Monnet

Information Retrieval

Rapport de projet

Idriss BENGUEZZOU
Ghilas MEZIANE

02/02/2024

Table des matières

1 Introduction

Ce projet a pour but de mettre en pratique les notions vues en cours et en TD. Il s'agit de réaliser un système de recherche d'information sur un corpus de documents, en utilisant les méthodes vues en cours. Le corpus de documents est constitué de 9804 documents, et les requêtes sont au nombre de sept.

L'idée est de répondre aux exigences spécifiques de compétitions comme l'INEX, qui imposent des standards stricts pour évaluer les performances des systèmes de recherche d'information. Notre rapport détaillera notre démarche, les choix méthodologiques, les paramétrages utilisés, et les résultats obtenus, en mettant particulièrement l'accent sur notre démarche de recherche de la meilleure configuration possible pour notre système de recherche d'information.

2 Méthodologie

Nous avons choisi de travailler avec le langage de programmation Python, au vu de sa popularité et de sa richesse en bibliothèques pour le traitement de texte. En outre, aucune bibliothèque n'a été utilisée pour la recherche d'information, l'indexation, ou le calcul de similarité. Nous allons détailler notre démarche pour chaque étape du projet.

2.1 Prétraitement des données

Lors de nos expérimentations, nous avons constaté que l'étape de prétraitement des données est cruciale pour la qualité des résultats obtenus. En effet, tout au long de nos expérimentations, nous avons constaté que les résultats obtenus étaient très sensibles à la qualité du prétraitement des données. Avec un prétraitement de qualité, nous avons pu améliorer notre score de MaGP de 0.198 à 0,2512.

Tout d'abord, nous avons utilisé la bibliothèque `nltk` pour le prétraitement des données. Nous avons utilisé la fonction `word_tokenize` pour la tokenisation des documents, et la fonction `PorterStemmer` pour la racinisation des mots. Nous avons également utilisé la liste de stopwords fournie en cours pour élimer les mots vides.

2.1.1 Prétraitement des données initial

Notre implémentation initiale des étapes de prétraitement des données a été la suivante :

1. Tokenisation
2. Mise en minuscule
3. Racinisation des mots (stemming)

4. Élimination des mots vides (suppression des stopwords)
5. Élimination de la ponctuation
6. Élimination des nombres
7. Élimination des caractères spéciaux
8. Élimination des mots chaines de caractères vides

Ces étapes de prétraitement des données était réaliser à chaque fois que nous devions indexer les documents. Nous verrons plus tard que par soucis d'optimisation des temps, nous avons choisi de réaliser ces étapes de prétraitement des données une seule fois, et de sauvegarder les résultats dans une nouvelle collection. Nous avons nommé cette première méthode de prétraitement des données : `Processing Reference`, car elle est réalisée à chaque fois que nous devons indexer les documents. Voici les statistiques de notre collection après cette première méthode de prétraitement des données :

Statistiques	Valeur
Taille moyenne des documents	654.008
Taille moyenne des mots	8.391
Nombre de mots unique	210,232
Nombre de mots total	6.41189e+06

TABLE 1 – Statistiques de la collection

Ces valeurs sont obtenues après avoir réalisé les étapes de prétraitement des données décrites ci-dessus. Elles nous ont permis d'avoir une référérence pour la suite de nos expérimentations, et de comparer les résultats obtenus avec d'autres méthodes de prétraitement des données. Une run de notre système de recherche d'information avec ces valeurs nous a donné un score de MaGP de 0.198.

2.1.2 Optimisation du Prétraitement des Données : Expérimentations avec un Ordre Modifié des Étapes

Suite à quelque expérimentations, nous avons constaté plusieurs choses qui nous ont poussé à modifier l'ordre des étapes de prétraitement des données. Nous avons remarqué que l'étape de `tokenisation` ne fonctionnait pas correctement par moment, par exemple : En tokenisant la phrase "World Ltd. World Ltd." nous obtenions ["World", "Ltd", ".", "World", "Ltd."] au lieu de ["world", "ltd", ".", "world", "ltd", "."]. En effet, en menant quelques recherches, nous avons remarqué que la tokenisation pouvais ne pas fonctionner correctement avec les abréviations, les mots vides, et les mots composés. Nous avons donc décidé de modifier l'ordre des étapes de prétraitement des données, et de réaliser la tokenisation après l'élimination de la ponctuation, des nombres, et des caractères spéciaux.

De plus, en parcourant notre liste de mots vides (stopwords), nous avons remarqué que tout les mots présent dans cette liste était en minuscule. Donc il était nécessaire de mettre en minuscule les mots avant de chercher à éliminer les mots vides. C'est à ce moment que nous avons remarqué que l'étape de racinisation des mots (stemming) était réalisée avant l'étape de suppression des mots vides. Ce qui fait que les mots vides n'étaient pas éliminés correctement. Nous avons donc décidé de réaliser la racinisation des mots après l'élimination des mots vides.

Notre nouvelle méthode de prétraitement des données est la suivante :

1. Élimination de la ponctuation
2. Élimination des nombres
3. Élimination des caractères spéciaux
4. Tokenisation
5. Mise en minuscule
6. Élimination des mots vides (suppression des stopwords)
7. Racinisation des mots (stemming)
8. Élimination des mots chaines de caractères vides

Nous avons nommé cette nouvelle méthode de prétraitement des données : `Processing Reference Réarrangé`. Voici les statistiques de notre collection après cette nouvelle méthode de prétraitement des données :

Statistiques	Valeur
Taille moyenne des documents	628.267
Taille moyenne des mots	9.227
Nombre de mots unique	226,564
Nombre de mots total	6.15953e+06

TABLE 2 – Statistiques de la collection

Nous pouvons tout de suite remarquer que la taille moyenne des documents a légèrement diminué, et que la taille moyenne des mots a légèrement augmenté. Le vocabulaire de notre collection a également augmenté. De plus nous avons un total de mots légèrement inférieur à la première méthode de prétraitement des données, environ 25,000 mots de moins.

Pour évaluer l'amplitude de l'impact de cette nouvelle méthode de prétraitement des données, nous avons décidé d'observer la fréquence des mots de nos requêtes avant et après cette nouvelle méthode.

	actor	algorithm	analysi	benefit	exclus	film	health	hill	inform	learn	link	machin
Processing Reference	7086	35062	8085	4671	1888	16595	11391	4513	17440	6003	12090	9122
Processing Reference Réarrangé	7113	35326	8164	4716	1891	16708	11411	4526	0	6028	12306	9160
Différence	27	264	79	45	3	113	20	13	-17440	25	216	38

TABLE 3 – Fréquence des mots des requêtes (Partie 1)

	model	mutual	network	oil	oliv	oper	probabilist	rank	retriev	score	supervis	system	web
Processing Reference	15337	2708	17820	14270	3023	16591	1230	3643	13560	4179	588	38683	20128
Processing Reference Réarrangé	15422	2711	17968	14409	3027	16622	1229	3680	13570	4198	589	39351	20188
Différence	85	3	148	139	4	31	-1	37	10	19	1	668	60

TABLE 4 – Fréquence des mots des requêtes (Partie 2)

Nous avons au total -15393 mots sur les mots de la requête. Nous remarquons immédiatement que le mot "information" (inform) à disparu de notre collection. Cela est dû à l'étape de suppression des mots vides (stopwords), qui a éliminé le mot "information" de notre collection. Ceci confirme que nos étapes de prétraitement sont bien réalisées

Après avoir retiré le mot "information" de notre list de mots vides (stopwords), nous avons un total de 2119 mots en plus sur les mots de la requête.

Voici une nouvelle fois les statistiques de notre collection après cette nouvelle méthode de prétraitement :

Statistiques	Valeur
Taille moyenne des documents	650.929
Taille moyenne des mots	8.42026
Nombre de mots unique	211,770
Nombre de mots total	6.3817e+06

TABLE 5 – Statistiques de la collection

Nous récupérons maintenant environs 150,000 mots en plus avec une taille moyenne des mots légèrement inférieure. En théorie, nous devrions obtenir de meilleurs résultats avec cette nouvelle méthode, donc en ce basant sur cette hypothèse, nous avons décider d'améliorer encore un peu plus notre méthode de prétraitement des données.

2.1.3 Optimisation du Prétraitement des Données : Expérimentations avec l'Élimination des Mots Rares

Nous avions remarqué que le vocabulaire de notre collection était très grand, nous avions environs 200,000 mots uniques. Cependant, en observant les mots de notre vocabulaire, nous nous sommes rendu compte que beaucoup de mots avait une fréquence très faible ou ne possédaient aucun sens. Nous avons donc décidé de créer une liste de mots que nous avons nommé `outliers`, qui contient les mots qui apparaissent moins de 5 fois dans notre collection et les mots ayant une longueur inférieure à 3 caractères. Nous avons ensuite éliminé ces mots de notre collection.

Voici les statistiques de notre collection après cette nouvelle méthode de prétraitement des données :

Statistiques	Valeur
Taille moyenne des documents	580.832
Taille moyenne des mots	6.68121
Nombre de mots unique	40,763
Nombre de mots total	5.69448e+06

TABLE 6 – Statistiques de la collection

A l'aide de cette nouvelle méthode notre vocabulaire est maintenant constitué de 40,763 mots uniques, avec une taille moyenne des mots de 6.68121. Nous avons donc réduit notre vocabulaire de plus de 150,000 mots. Nous avons également une taille moyenne des documents de 580.832 mots, ce qui est une réduction de 70 mots par rapport à la méthode précédente.

3 Méthodes et Paramétrages

Nous avons utilisé plusieurs méthodes et modèles pour la recherche d'information et avons réalisé plusieurs expérimentations pour essayé de trouver la meilleure configuration possible. Nous présentons dans cette section les différentes méthodes et modèles que nous avons utilisé, ainsi que les paramétrages que nous avons utilisé, et les résultats obtenus.

3.1 Méthodes de Recherche d'Information

Tout au long de ce projet, nous avons recherché à trouver et optimiser la meilleure méthode de recherche d'information. N'ayant pas la capacité d'évaluer de notre côter

nos modèles, nous avons mis en place quelque outils pour nous permettre de comparer des runs entre elles.

Nous avons notamment utilisé des mesures de corrélation pour comparer les résultats obtenus, ainsi nous étions en mesures de savoir si des changements dans le code modifiaient les runs produites. Cette mesure de corrélations ne nous à pas permis d'évaluer nos modèles, mais nous à permis de comparer une run "référérence" en l'occurrence une run baseline, avec d'autres runs pour voir si les changements celles ci avait un classement similaires ou pas.

En ce qui concerne les runs éléments, nous avons calculé le nombre de balises ciblées retourné dans les runs. Ainsi, si une balise ciblée était très peu présente dans les runs, nous pouvions en déduire que cette balise ne comportait pas beaucoup d'information, et que nous devions peut-être la retirer de notre liste de balises ciblées.

3.1.1 Méthodes de Recherche d'Information : Smart LTN

Le poids TF-IDF (*Term Frequency-Inverse Document Frequency*) d'un terme dans un document est déterminé par la formule suivante :

$$\text{TF-IDF}(i, d) = (1 + \log_{10}(\text{TF}(i, d))) \times \text{IDF}(i) \tag{1}$$

où :
- $\text{TF}(i, d)$ représente la fréquence du terme i dans le document d.
- $\text{IDF}(i)$ est la fréquence inverse du document pour le terme i.

Les composants individuels sont calculés de la manière suivante :

$$\text{IDF}(i) = \log_{10}\left(\frac{\text{Taille de la Collection}}{\text{Fréquence du Terme}(i)}\right) \tag{2}$$

$$\text{TF}(i, d) = \text{Fréquence du Terme}(i, d) \tag{3}$$

Dans le cadre de notre système, l'IDF et le TF sont calculés en tenant compte des balises ciblées. Ci-dessous quelques résultats obtenus sur notre run baseline.

Run Name	**MAGP Value**	**P[0, 1]**
..._5_ltn_article_stop670_porter	0.1687	0.3835
..._8_ltn_article_stop670_nostem	0.1699	0.3996
..._11_ltn_article_nostop_porter	0.1515	0.3265
..._14_ltn_article_nostop_nostem	0.1699	0.3996

TABLE 7 – Résultats de la méthode Smart LTN

Nous pouvons remarquer que le fait de ne pas éliminer les mots vides (stopwords) a un impact négatif sur les résultats obtenus. En effet, les runs 8 et 14 ont obtenu les

meilleurs résultats, avec un score de MaGP de 0.1699, et un P[0, 1] de 0.3996. Nous avons également remarqué que la racinisation des mots (stemming) n'a pas d'impact significatif sur les résultats obtenus. En effet, les runs 8 et 14 ont obtenu les mêmes résultats, alors que le run 8 a été réalisé avec la racinisation des mots, et le run 14 sans la racinisation des mots.

3.1.2 Méthodes de Recherche d'Information : Smart LTC

Le schéma de pondération LTC (Logarithmique-Term Frequency and Cosine Normalization) optimise la précision des recherches en ajustant les poids des termes selon leur fréquence et la longueur des documents. Voici les étapes clés de notre implémentation :

1. **Calcul du Poids LTC** : Utilise la fréquence logarithmique du terme (L), la fréquence inverse du document (T), et la normalisation cosinus (C) pour équilibrer l'importance des termes entre documents de longueurs variées.
2. **Fréquence Logarithmique du Terme (L)** :

$$L(i,d) = 1 + \log_{10}(\mathrm{TF}(i,d)) \tag{4}$$

   $\mathrm{TF}(i,d)$ représente la fréquence du terme i dans le document d.
3. **Fréquence Inverse du Document (T)** :

$$T(i) = \log_{10}\left(\frac{N}{\mathrm{DF}(i)}\right) \tag{5}$$

   N est le nombre total de documents, et $\mathrm{DF}(i)$ le nombre de documents contenant le terme i.
4. **Normalisation Cosinus (C)** :

$$C(d) = \frac{1}{\sqrt{\sum_{i \in d}(L(i,d) \times T(i))^2}} \tag{6}$$

5. **Poids Final LTC** :

$$\mathrm{LTC}(i,d) = (L(i,d) \times T(i)) \times C(d) \tag{7}$$

L'implémentation de ce schéma a permis à notre système une évaluation équilibrée des termes en fonction de leur fréquence et de la longueur des documents, améliorant la pertinence des résultats de recherche.

3.1.3 Méthodes de Recherche d'Information : Smart LNU

Le schéma de pondération LNU (Logarithmic Normalization and Unique-term adjustment) ajuste le poids des termes en fonction de leur fréquence, de la longueur du document, et du nombre de termes distincts. Le processus est résumé comme suit :

Run Name	MAGP Value	P[0, 1]
BengezzouIdrissMezianeGhilas_6_ltc_article_stop670_porter	0.0834	0.22818
BengezzouIdrissMezianeGhilas_9_ltc_article_stop670_nostem	0.0699	0.1987
BengezzouIdrissMezianeGhilas_12_ltc_article_nostop_porter	0.0668	0.1930
BengezzouIdrissMezianeGhilas_15_ltc_article_nostop_nostem	0.0699	0.1987

TABLE 8 – Résultats de la méthode Smart LTC

1. **Fréquence Logarithmique Normalisée (LNN)** :

$$\text{LNN}(i, d) = 1 + \log_{10}(\text{TF}(i, d)) \tag{8}$$

où $\text{TF}(i, d)$ est la fréquence du terme i dans le document d.

2. **Normalisation par la Longueur** :

$$\text{LN}(d) = 1 + \log_{10}\left(\frac{\text{Length}(d)}{\text{AVDL}}\right) \tag{9}$$

ajuste le poids selon la longueur du document d par rapport à la longueur moyenne des documents AVDL.

3. **Ajustement Basé sur les Termes Distincts** :

$$\text{Adjustement}(d) = (1 - \text{slope}) + \text{slope} \times \frac{\text{NTD}(d)}{\text{Pivot}} \tag{10}$$

où $\text{NTD}(d)$ est le nombre de termes distincts dans d.

4. **Poids Final** :

$$\text{Weight}(i, d) = \frac{\text{LNN}(i, d)}{\text{LN}(d)} / \text{Adjustement}(d) \tag{11}$$

combinant les éléments ci-dessus.

Voici les scores MaGP que nous avons obtenu en essayant de faire varier le slope de 0.1 à 1.1 [4].

Comme nous pouvons le remarquer ici, le MaGP est bien loin des scores que nous avons pu atteindre en utilisant d'autres modèle de pondération. Après révision du code, nous nous sommes aperçus d'une erreur au niveau du calcul de l'ajustement dans la formule LNU. Les poids étaient calculés de la manière suivante :

```
adjustement = (1 - self.slope) + pivot + (self.slope * nt_d)
    weight = (lnn / length_normalization) / adjustement
```

Cette erreur a conduit à un calcul inexact de l'ajustement, impactant directement le poids final attribué à chaque terme dans les documents, et cette mauvaise estimation a affecté la capacité du modèle à distinguer efficacement les documents pertinents des non pertinents, ce qui se reflète dans une baisse de la précision moyenne (MaGP)

3.1.4 Méthodes de Recherche d'Information : BM25

La méthode de pondération BM25, également connue sous le nom de Best Matching 25, est une technique couramment utilisée en recherche d'information pour attribuer des poids aux termes dans un index inversé. Cette méthode prend en compte la fréquence du terme dans un document, la fréquence du terme dans l'ensemble de la collection, et la longueur du document.

Le calcul du poids BM25 pour un terme dans un document est défini par la formule suivante :

$$\text{BM25}(i, d) = \frac{\text{TF} \times (\text{k1} + 1)}{\text{k1} \times \left((1 - \text{b}) + \text{b} \times \left(\frac{\text{DL}}{\text{AVDL}}\right)\right) + \text{TF}} \times \text{IDF} \tag{12}$$

où :

- TF est la fréquence du terme dans le document.
- DL est la longueur du document.
- AVDL est la longueur moyenne des documents dans la collection.
- IDF est la fréquence inverse du document pour le terme.
- k1 et b sont des paramètres de réglage qui contrôlent l'impact de la fréquence du terme et de la longueur du document, respectivement.

La mise en œuvre de la méthode BM25 dans notre système de recherche d'information est réalisée par la classe `BM25Weighting`, héritant de la classe abstraite `WeightingStrategy`. Cette classe propose une méthode `calculate_weight` qui construit l'index inversé pondéré en utilisant la pondération BM25. Les paramètres k1 et b sont configurables lors de l'instanciation de la classe `BM25Weighting`, pour permettre d'expérimenter avec différentes valeurs de ces paramètres.

Les résultats obtenus avec cette méthode seront présentés dans la section suivante, soulignant son impact sur les performances du système de recherche d'information.

Run Name	**MAGP Value**	**P[0, 1]**
..._7_bm25_article_stop670_porter_k1_b0.5	0.2368	0.5157
..._10_bm25_article_stop670_nostem_k1_b0.5	0.2364	0.5634
..._13_bm25_article_nostop_porter_k1_b0.5	0.1924	0.4325
..._16_bm25_article_nostop_nostem_k1_b0.5	0.2332	0.5537

Table 9 – Résultats de la méthode BM25

Nous remarquons encore une fois que les runs ayant les meilleurs résultats sont ceux qui ont été réalisés avec la racinisation des mots (stemming), et sans l'élimination des mots vides (stopwords).

De plus, en comparaisons avec les résultats obtenus avec la méthode Smart LTN et la méthode Smart LTC, BM25 a obtenu les meilleurs résultats, avec un score de MaGP de 0.2368 sur la run 7, et un P[0, 1] de 0.5634 sur la run 10.

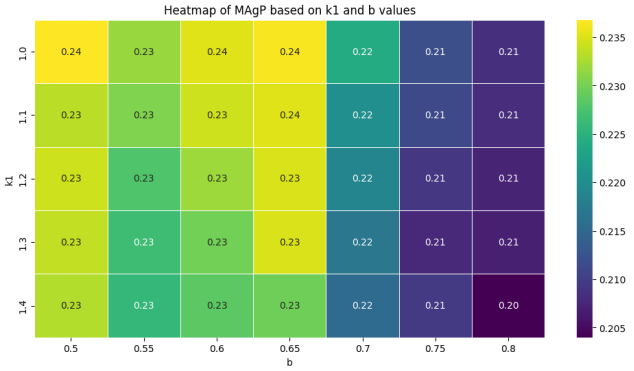

TABLE 10 – Evolution de MaGP en fonction de k1 et b

Nous pouvons remarquer que notre score de MaGP est maximisé pour des valeurs de b assez faible (entre 0.5 et 0.65) et des valeurs de k1 entre 1 et 1.1.

Nous pouvons remarquer très clairement sur le graphique que notre score de MaGP est maximisé avec une valeur de b égale à 0.5. En effet, sur notre interval de recherche pour k1, nous avons un score élevé et assez stable pour des valeurs de k1 entre 1 et 1.4.

Nous avons également voulu observer si les valeurs de k1 et b qui maximisent notre score de MaGP maximisent également notre score de P[0, 1]. Voici les résultats obtenus :

Nous remarquons tout de suite que les valeurs maximisant notre score de MaGP ne maximisent pas notre score de P[0, 1]. En effet, nous avons un score de P[0, 1] maximisé pour des valeurs de b entre 0.65 et 0.75, et des valeurs de k1 entre 1.0 et 1.3.

Nous savons donc que les valeurs permettant de maximiser notre score de MaGP ne maximisent pas notre score de P[0, 1].

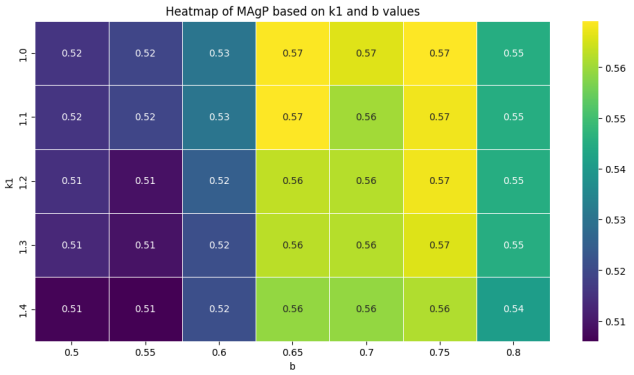

TABLE 11 – Evolution de P[0, 1] en fonction de k1 et b

Voir 5 la courbe de rappel-précision pour notre meilleur run article avec BM25.

En nous concentrant sur la courbe de rappel-précision, nous pouvons observer que notre modèle BM25 a une précision assez faible pour des valeurs de rappel assez élevées.

Nous avons par la suite essayé d'évaluer notre modèle BM25 en essayant de cibler les balises. En ciblant uniquement la balise `body`, nous avons obtenu notre meilleur score de MaGP, avec un score de 0.2512 et un P[0, 1] de 0.5136.

Concernant les résultats obtenus avec la méthode BM25 en ciblant les balises, nous avons obtenu les meilleurs résultats en associant systématiquement la balise `body` ou `article` avec d'autres balises plus spécifiques, comme `header` ou `section`.

Run Id	Granularity	MAPGP Score	P[0, 1] Score
92	bdy, header	0.1685	0.3731
88	article, event, title	0.1667	0.3299
18	article, header	0.1621	0.3711
20	article, sec	0.1499	0.3817
93	bdy, header, sec	0.1454	0.3708

TABLE 12 – Résultats de la méthode BM25 avec ciblage de balises

De plus, lors de nos expérimentations nous avons essayé deux approche différentes pour le calcul du DF (Document Frequency). Nous avons calculé le DF en prenant en compte les chemins des balises, et en ne prenant pas en compte les chemins des balises. Nous avons constaté que le fait de ne pas prendre en compte les chemins des balises a un impact très négatif sur les résultats obtenus. En effet, une simple run avec la balise `body` nous a donné un score de MaGP de 0,0752 et une precision de 0,3379 (voir run `6_bm25_bdy_stop671_porter_k1_b0.5` practice 5v4).

En ne prenant pas en compte les chemins des balises, nous sommes revenus à des run `body` et `article` très similaires.

3.1.5 Méthodes de Recherche d'Information : BM25F$_w$

La méthode de pondération $BM25F_w$ étend la méthode BM25 traditionnelle en intégrant une approche pondérée des champs pour améliorer la précision de la recherche d'information. Cette technique est particulièrement efficace pour les documents structurés, où les termes peuvent avoir une importance différente selon leur contexte dans le document.

La formule de base pour le calcul du poids $BM25F_w$ pour un terme dans un document multi-champs est :

$$BM25F_w(i,d) = \sum_{f \in F} w_f \times \frac{\mathrm{TF}_f \times (\mathrm{k1}+1)}{\mathrm{k1} \times \left((1-\mathrm{b}) + \mathrm{b} \times \left(\frac{\mathrm{DL}_f}{\mathrm{AVDL}_f}\right)\right) + \mathrm{TF}_f} \times \mathrm{IDF} \qquad (13)$$

La différence principale réside dans l'introduction de poids de champ (w_f) dans BM25F_w, permettant d'ajuster l'importance relative de chaque champ dans le calcul de la pertinence. Cette approche permet à BM25F_w de reconnaître que certains termes peuvent avoir plus de signification lorsqu'ils apparaissent dans certaines parties du document, comme le titre, par rapport à leur présence dans le corps du texte.

Nous avons évalué les performances de notre modèle BM25F_w en utilisant différentes combinaisons des paramètres α, β, et γ, avec $k1 = 1.2$ et $b = 0.75$.

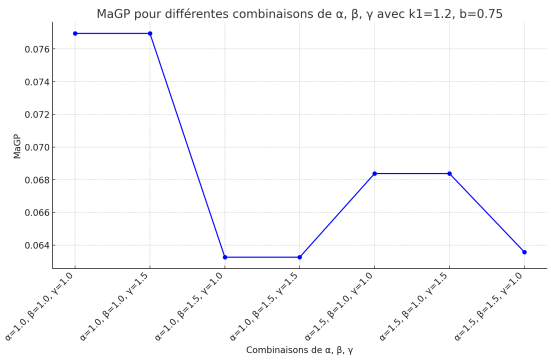

FIGURE 1 – Résultats MaGP BM25FW sous différents paramètres

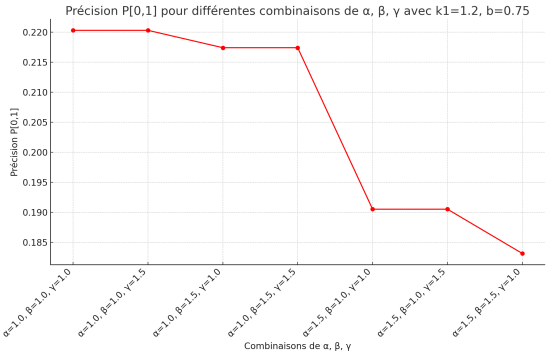

FIGURE 2 – Résultats de précision P[0,1] pour BM25FW sous différents paramètres

Analyse des résultats

Concernant la Mean Average Precision (MaGP), nous avons constaté des variations similaires en fonction des paramètres α, β, et γ. Les configurations initiales ($\alpha = 1.0, \beta = 1.0$) ont montré une performance robuste indépendamment de la valeur de γ, tandis que l'augmentation de α et β a conduit à une diminution de MaGP, particulièrement pour ($\alpha = 1.5, \beta = 1.5, \gamma = 1.0$).

Les meilleures performances en termes de précision P[0,1] ont été observées pour les configurations ($\alpha = 1.0, \beta = 1.0, \gamma = 1.0$) et ($\alpha = 1.0, \beta = 1.0, \gamma = 1.5$), avec une précision de 0.22. La diminution de la précision pour des valeurs plus élevées de β et γ suggère que surpondérer ces champs par rapport à α n'améliore pas la qualité des premiers résultats de recherche. La précision la plus faible a été enregistrée pour la combinaison ($\alpha = 1.5, \beta = 1.5, \gamma = 1.0$), soulignant l'importance d'un équilibre approprié entre les paramètres pour maximiser l'efficacité du modèle. Dans notre cas, l'application du modèle BM25F_w sur des balises d'articles a produit un Mean Average Precision (MaGP) différent de celui observé lors de l'utilisation du modèle BM25 classique. Cette différence suggère que le choix des balises sur lesquelles le modèle est appliqué inluence de manière significative les résultats obtenus, car BM25F_w est conçu pour tirer parti de la structure des documents en attribuant des poids différents aux termes selon leur emplacement dans différents champs ou balises du document.

3.1.6 Méthodes de Recherche d'Information : BM25F$_r$

Dans le cadre de ce projet nous avons aussi eu l'occasion d'implémenter une autre méthode de pondération : $BM25F_r$. La méthode de pondération $BM25F_r$, proposée par Robertson, est une variante du modèle BM25 classique qui est spécifiquement conçue pour améliorer la recherche d'informations dans des documents structurés. Tout comme $BM25F_w$, $BM25F_r$ étend le modèle BM25 en prenant en compte la structure des documents, mais avec une approche différente focalisée sur la représentation et la pondération des termes en fonction de leur pertinence dans divers champs du document.

Contrairement à BM25F_w, qui ajuste l'importance des champs dans le calcul de la pertinence à travers des poids statiques w_f, BM25F_r adopte une approche de pondération dynamique. Cette méthode ajuste la contribution de chaque champ en fonction de ses caractéristiques propres, utilisant des facteurs de normalisation et de pondération variables. Cela permet une évaluation plus précise et adaptative de la pertinence des termes selon le contexte spécifique de chaque champ, suivant les principes énoncés par Robertson. La formule de pondération pour BM25F_r est représentée comme suit :

$$BM25F_r(t, D) = IDF(t) \times \sum_{f \in F} \left(\frac{(k_1 + 1) \times TF_{f,t,D}}{K_{f,D} + TF_{f,t,D}} \right) \times W_f \tag{14}$$

Ci-dessous les résultats obtenus par nos runs $BM25F_r$:

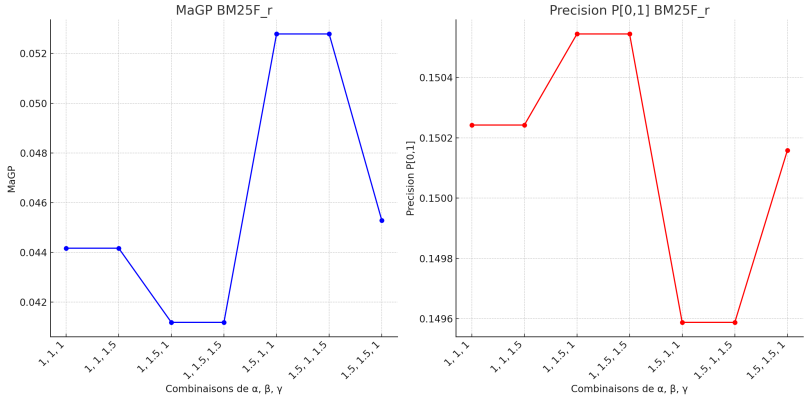

FIGURE 3 – MaGP and Precision - BM25F$_r$

Encore une fois, nos résultat sont très éloignés d'une run BM25 "classique" pour des paramètres à 1. Cela est très probablement du au choix des balises pour réaliser nos runs, car en effet, en effectuant des tests sur des collection beaucoup plus petite nous obtenions des résultats identiques à ceux de BM25. Malheureusement, nous n'avons pas eu le temps de réaliser des expérimentations plus poussées pour sélectionner les bonnes balises. Cependant, nous avons quand même pu observer qu'en sélectionnant les balises `body` et `article` nous obtenions les même résultats que pour BM25.

3.1.7 Méthodes de Recherche d'Information : DeepLearning

Dans cette section, nous explorons l'utilisation de méthodes de recherche d'information basées sur le Deep Learning. Notre démarche s'est concentrée sur l'exploration de modèles pré-entraînés et l'évaluation de leur performance dans le contexte de la recherche d'information (IR).

Nous avons entrepris des recherches approfondies dans la littérature liée à l'IR et au Deep Learning, identifiant de nombreuses ressources pertinentes. Souhaitant évaluer ces méthodes, nous avons choisi d'effectuer plusieurs runs, en utilisant des modèles pré-entraînés disponibles.

Parmi les documents que nous avons trouvés particulièrement intéressants, nous mettons en avant le papier intitulé "BEIR : A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models" (https://openreview.net). Ce document propose un benchmark de plusieurs modèles sur divers ensembles de données. De plus, les auteurs ont rendu leur code accessible sur Github.

Pour nos expérimentations, nous avons utilisé le code mis à disposition par les auteurs et généré des runs pour évaluer les performances de ces modèles sur notre collection de documents.

- **msmarco-distilbert-base-tas-b** : Utilisation de DistilBERT avec une fonction de similarité cosinus. Les embeddings de la requête et du document sont générés dans le même espace vectoriel.
- **msmarco-roberta-base-ance-firstp** : Utilisation de Sentence-BERT avec une fonction de produit scalaire. Intègre ANCE, une approximation d'ANN (Approximate Nearest Neighbors), pour les embeddings multilingues.
- **facebook/dpr-question_encoder-multiset-base** : Utilisation de DPR (Deep Passage Retrieval) avec des encodeurs de question et de contexte. La fonction de score est basée sur le produit scalaire pour la récupération dense de passages.

La librairie BEIR fournit d'autres modèles et techniques, telles que le Re-ranking, qui permet de réordonner les résultats d'une recherche initiale en utilisant un modèle de Deep Learning. Les auteurs proposent 5 techniques différentes :

- **Lexical Retrieval :** Avec un BM25 associé à ElasticSearch ou Anserini.
- **Sparse Retrieval :** Utilisation de modèles DeepCT, SPARTA, et DocT5query.
- **Dense Retrieval :** Utilisation de modèles DPR, ANCE, TAS-B, et GenQ.
- **Late-Interaction :** Utilisation de ColBERT pour l'agrégation des interactions tardives.
- **Re-Ranking Model :** Utilisation de modèles comme BM25 + CE, qui réordonne les résultats d'une recherche initiale avec un modèle de distillation des connaissances.

Les résultats obtenus avec ces modèles était très faible sur notre collection. Il est possible que les transformers utilisés pour ces modèles ne soient pas adaptés à notre collection de documents.

4 Performances & Optimisations

Dans cette section, nous présentons les efforts déployés pour améliorer les performances et optimiser le code, réduisant ainsi les temps d'exécution. Nous avons adopté les principes de la Programmation Orientée Objet (POO) pour structurer notre code de manière plus efficace.

En ce qui concerne les structures de données, nous avons entrepris des optimisations sur des éléments clés tels que l'index inversé, la fréquence des termes, la fréquence des documents, etc. L'objectif principal était d'optimiser ces structures pour favoriser les accès directs aux données, atteignant ainsi une complexité temporelle de O(1) dans de nombreux cas.

Par ailleurs, afin d'optimiser l'efficacité des recherches, nous avons suivi les conseils donnés en classe. Une des approches adoptées a été de n'indexer que les mots présent dans les requêtes, réduisant ainsi la taille de l'index et améliorant les performances globales du système. Ces ajustements ont été réalisés dans le but d'obtenir des temps d'exécution plus rapides et une utilisation plus efficace des ressources.

4.1 Réduction significative du Temps d'Indexation

Nous avons réalisé des avancées majeures en réduisant drastiquement le temps d'indexation grâce à un prétraitement des données optimisé. Initialement, le temps d'indexation pour une granularité comprenant ['.//bdy', './/header', './/sec', './/p'] était de l'ordre de plusieurs dizaines de secondes. Après optimisation, nous avons obtenu les temps d'indexation suivants :

- Avec les balises ['.//bdy', './/header', './/sec', './/p'], le temps a été réduit à **5.62 secondes**, contre plus de 10 secondes précédemment.
- L'ajout de './/list' a légèrement augmenté le temps à **5.99 secondes**, démontrant l'efficience des optimisations malgré l'augmentation de la complexité.
- L'inclusion de './/entry' a porté ce temps à **6.76 secondes**, toujours bien inférieur aux performances initiales.

4.2 Impact sur les Résultats de Recherche

Outre l'amélioration des performances d'indexation, ces optimisations ont eu un impact positif sur la qualité des résultats de recherche. En éliminant plus efficacement les termes non pertinents et en se concentrant sur les termes significatifs grâce à un meilleur prétraitement, nous avons vu une amélioration notable de la précision et de la pertinence des résultats.

5 Conclusion

Ce projet nous a permis de mettre en pratique les connaissances acquises en cours de Recherche d'Information. Nous avons exploré diverses méthodes et paramétrages pour optimiser un système de recherche d'information sur un corpus de documents. Notre travail s'est concentré sur l'implémentation de modèles basés sur les pondérations TF-IDF, BM25, $BM25F_w$, et $BM25F_r$, ainsi que sur l'expérimentation avec des techniques de prétraitement des données pour améliorer les performances.

Un aspect crucial de notre travail a été l'optimisation de la structure de données et des performances de notre système. En adoptant une approche méthodique pour améliorer les structures de données telles que l'index inversé, nous avons pu réduire significativement les temps d'indexation et de pondération. Ces optimisations ont permis de rendre notre système plus efficient, en réduisant la charge de calcul et en accélérant les temps d'indexation et de weighting.

Les résultats obtenus ont montré l'importance cruciale du prétraitement des données et du choix judicieux des paramètres pour les modèles de pondération. Nous avons constaté que des ajustements fins des paramètres, comme ceux de BM25 ($k1$ et b), ainsi que la sélection stratégique des balises pour $BM25F_w$ et $BM25F_r$, peuvent avoir un impact significatif sur la qualité des résultats de recherche.

Il est apparu que l'application de ces modèles sur des balises spécifiques peut influencer de manière significative les résultats, soulignant l'importance de comprendre la structure des documents pour une indexation et une recherche efficaces.

Dans le cadre de notre quête pour améliorer les performances de notre système de recherche d'information, nous avons également exploré l'application de techniques de Deep Learning. Reconnaissant le potentiel du Deep Learning pour capter les nuances sémantiques des termes et améliorer la pertinence des résultats de recherche, nous avons expérimenté avec des modèles pré-entraînés tels que BERT et ses variantes.

Malgré l'intérêt prometteur de ces modèles, l'application du Deep Learning à notre corpus spécifique a présenté des résultats mitigés. Il est probable que les modèles pré-entraînés que nous avons utilisés ne soient pas adaptés à notre collection de documents, soulignant la nécessité de sélectionner des modèles et des techniques adaptés à la nature spécifique de notre corpus.

En conclusion, ce projet a non seulement renforcé notre compréhension des principes fondamentaux de la recherche d'information mais a également mis en lumière les défis associés à l'optimisation des systèmes de recherche d'information. Les leçons apprises ici, notamment l'importance du prétraitement, le choix des paramètres, et la compréhension de la structure des documents, seront précieuses pour nos futurs travaux dans le domaine.

6 Annexes

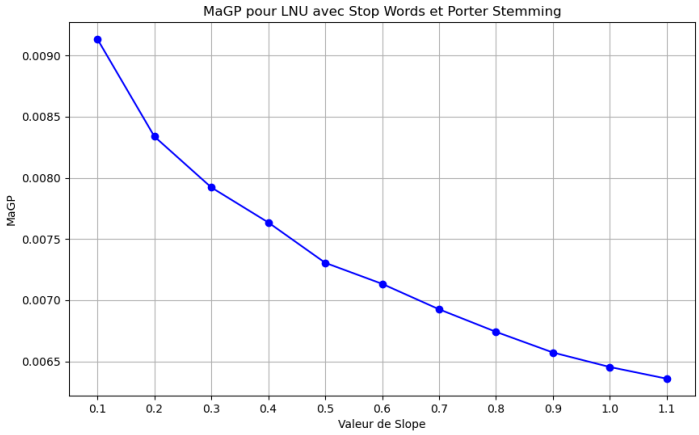

FIGURE 4 – MaGP pour LNU en fonction de différents slopes

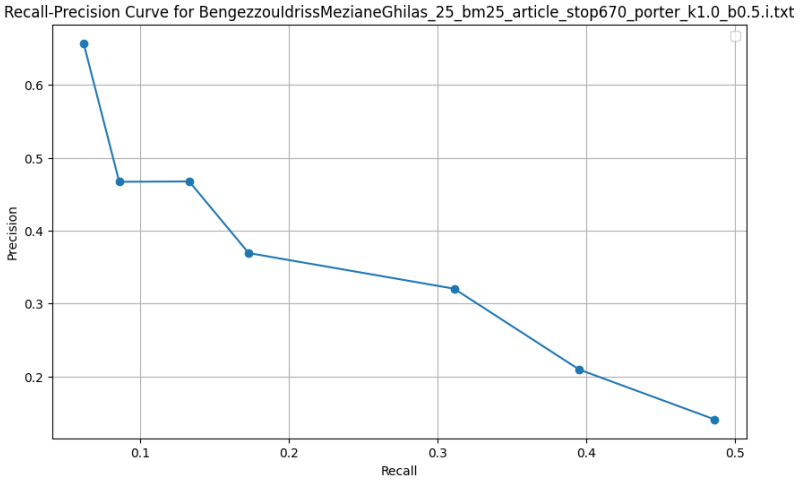

FIGURE 5 – Courbe de rappel-précision pour BM25