Lab01 Web Mining : Crawling, indexation et recherche de pages Web

Jade Gröli & Dalia Maillefer & David González León

Question 2.1 :

Pour notre crawler nous avons décidé de nous concentrer sur les pages Wikipédia en français. Nous avons décidé de récupérer sur ces pages le premier paragraphe de texte de chaque page, et de récupérer quelques informations du dictionnaire de schema.org présent sur chaque page :

- L'url
- Le titre (name)
- La headline (si elle existe)

Nous avons également récupéré le nombre de liens vers d'autres sites Wikipédia sur la page et avons stocké cette valeur numérique.
Après avoir extrait ces valeurs, nous continuons notre exploration sur toutes les pages Wikipédia présentes sur cette page, et ainsi de suite. Pour éviter d'avoir des doublons d'information, nous avons configuré Scrapy afin de ne visiter une page qu'une seule fois.

Question 2.3 :

Voici un exemple de query possible avec ElasticSearch :

```
GET /_search
{
  "query": {
    "multi_match" : {
      "query":    "this is a test", 1
      "fields": [ "subject", "message" ] 2
    }
  }
}
```

Pour faire des query simples sous ElasticSearch en recherchant uniquement dans certains champs, on peut spécifier les champs voulus dans le tableau après "fields".

Voici un exemple de query permettant de "boost" un champ spécifique :

```
GET /_search
{
  "query": {
    "multi_match" : {
      "query" : "this is a test",
      "fields" : [ "subject^3", "message" ] 1
    }
  }
}
```

Dans ce cas, le champ “subject” aura un poids plus élevé dans la pondération du score par ElasticSearch que le champ “message”.

Question 2.4 :

Question 2.4.1 :

On peut ajouter un champ “langue” à chaque élément de l’index, afin de pouvoir ensuite adapter nos query pour chaque langue.
Le désavantage de cette solution est qu’il faut préciser autant de query qu’il y a de langues différentes dans les pages répertoriées dans l’index. Il faut donc savoir à l’avance toutes les langues qu’on veut traiter.
L’avantage est qu’on a accès à plus d'informations, car on traite plus de pages.

Il faudrait donc adapter à chaque langue les queries en traduisant leur contenu en fonction de la langue de la page.

Question 2.4.2 :

La “fuzzy query” d’ElasticSearch est une requête qui permet de trouver dans des documents des termes proches du terme recherché. Ils utilisent pour cela la distance de Levenshtein. ElasticSearch va chercher tous les termes avec une distance au terme dans la query inférieure à un certain seuil et renvoyer tous ces résultats.

Plus il y a de variations sur le terme recherché, plus les performances seront impactées, étant donné que ElasticSearch doit comparer avec tous les termes suffisamment similaires. On peut imposer des limites sur la taille du terme ou la distance minimale pour essayer d’améliorer la performance, mais ceci peut avoir un impact sur la pertinence des résultats.

Une alternative à la requête floue est d'utiliser un analyseur personnalisé qui normalise les variations de noms avant l'indexation et la recherche. Par exemple, on peut utiliser un filtre phonétique qui transforme les noms en codes basés sur leur prononciation. Ainsi, les noms qui se prononcent de façon semblable auront le même code et seront trouvés par une requête simple. Cette approche peut améliorer la performance et la précision, mais elle nécessite une configuration plus complexe et peut ne pas couvrir tous les cas possibles.