université PARIS-SACLAY

M2 AIC

Recherche et Extraction d'Information

Projet : Recherche et Extraction d'Information

Table des matières

1 Introduction

Ce projet contient deux parties nommées “Tronc Commun” et “Options” :

— Le **tronc commun**, présenté à la section 3, doit être réalisé par tous les groupes.
— Chaque groupe choisira ensuite **une option** parmi celles présentées à la section 4.

L'objectif du tronc commun est de construire un moteur de recherche fonctionnant sur une base statique de documents de taille moyenne. Ce moteur sera implémenté en java et pourra reprendre l'ensemble des classes écrites au fil des TP. Les options sont des suites possibles de ce travail.

Les projets seront effectués en **trinômes** (on composera un ou deux quadrinômes si nécessaire). Le projet rendu, le rapport ainsi que la soutenance donneront lieu à la note finale du module (voir le site web pour le barême détaillé).

2 Le corpus

La collection de documents WEB-NEWS est composée de **1,55 million de documents**. Ces documents sont des articles de journaux de la presse généraliste francophone, parus en ligne entre mars 2013 et septembre 2015 et collectés grâce aux flux RSS des sites.

Cette collection est disponible sur le réseau dans ce répertoire :

`/public/rei/projetREI/`

contenant l'architecture suivante :

```
.
├── feed_list.txt
├── index/
│   ├── 2014
│   │   └── 01
│   │       ├── 20140101.xml
│   │       ├── 20140102.xml
│   │       └── ...
│   ├── 2015
│   │   └── ...
│   └── ...
├── subindex/
├── data/
│   ├── 2014
│   │   └── 01
│   │       └── 01
│   │           ├── 0033a7130ba8ebc87b43723c2c9d5a6a.html
│   │           ├── 00588c5c436279376bcf1d47351e3848.html
│   │           └── ...
│   ├── 2015
│   └── ...
└── text/
    ├── 2014
    │   └── 01
    │       └── 01
    │           ├── 0033a7130ba8ebc87b43723c2c9d5a6a.xml
    │           ├── 00588c5c436279376bcf1d47351e3848.xml
    │           └── ...
    └── 2015
```

Le fichier `feed_list.txt` contient la liste des flux utilisés, accompagné du thème du flux lorsqu'il est précisé (économie, sport, politique, etc.).

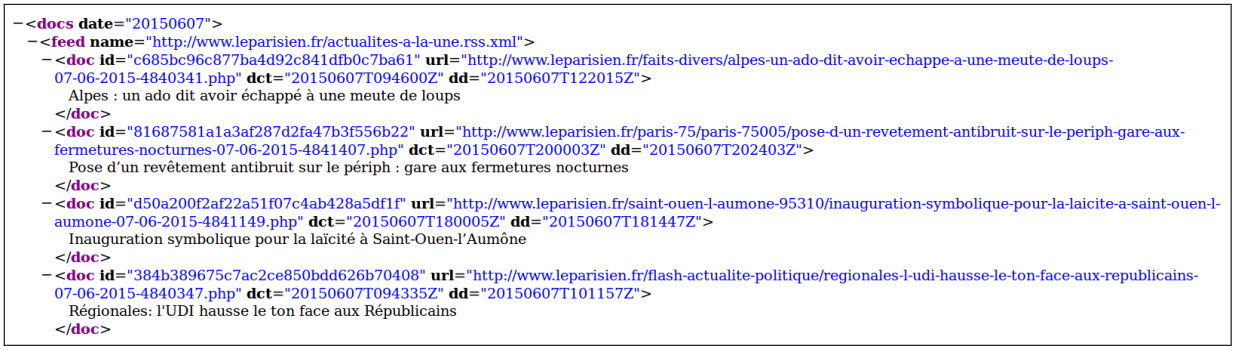

FIGURE 1 – Exemple de fichier d'index au format XML pour la collection WEB-NEWS.

Dans le répertoire `index`, vous trouverez des fichiers d'index au format XML, dont un extrait est présenté à la figure 1. On y trouve les informations suivantes :

- `<docs date="20150607">` : cette date est celle annoncée par les flux RSS qui répertorient les pages Web collectées. Souvent fausse ; vous préférerez utiliser l'attribut `dct` de chaque lien.
- `<feed name="...">` : l'URL du flux RSS utilisé pour collecter l'information. Si plusieurs flux contiennent la même page web, alors cette page sera indexée plusieurs fois.
- `<doc id="..."` : l'identifiant unique de la page web, permettant d'aller chercher le fichier correspondant dans le répertoire `data` (voir ci-dessous).
- `<doc url="..."` : l'URL de la page web téléchargée.
- `<doc dct="..."` : la date de création du document (DCT = Document Creation Time), estimée à partir d'une heuristique générant quelques erreurs, mais plus fiable que le champ `date`. La date est au format '`YYYYMMDDTHHMMSSZ`', ainsi, `20150607T094603Z` signifie le 7 juillet 2015 à 09h46 :03.
- `<doc dd="..."` : la date de téléchargement du document.

Le répertoire `subindex` suit la même architecture que l'`index` mais ne répertorie qu'un sous-ensemble d'environ 10 000 documents du corpus publiés entre janvier 2015 et avril 2015. De plus, la balise `feed` a été supprimée.

Dans le répertoire `data`, vous trouverez l'ensemble des pages web téléchargées. Le nom du fichier correspond à l'identifiant présent dans l'index.

Enfin, dans le répertoire `text`, ces mêmes pages ont été nettoyées, c'est-à-dire débarrassées des balises, des scripts, mais également de tout l'environnement non informatif : menus, publicités, commentaires, etc. Seul le contenu de l'article est conservé. Ce nettoyage a été opéré avec l'outil Boilerpipe [1] (qui peut commettre des erreurs) puis segmenté en phrases avec l'outil openNLP [2]. Le résultat est un fichier contenant uniquement du texte, avec une phrase par ligne.

Pour accéder à un document à partir de l'index.

- Pour accéder à la version HTML, si la date de téléchargement du document est `YYYYMMDDTXXXXXXZ` et si l'identifiant du document est `ID`, alors le fichier recherché est
  `data/YYYY/MM/DD/ID.html`
- Pour accéder à la version nettoyée, si la date de création du document (`dct`) du document est `YYYYMMDDTXXXXXXZ` et si l'identifiant du document est `ID`, alors le fichier recherché est
  `text/YYYY/MM/DD/YYYYMMDD_ID.xml`

1. `https://code.google.com/p/boilerpipe/`
2. `https://opennlp.apache.org/`

3 Tronc commun : moteur de recherche

Dans cette partie commune à tous les groupes, vous utiliserez les documents répertoriées dans le répertoire `subindex`, soit environ **10 000 documents** du corpus publiés entre janvier 2015 et avril 2015. En utilisant les travaux effectués en TP et les techniques de passage à l'échelle vues en cours (gestion de la mémoire, de l'espace disque, etc.), vous construirez un moteur de recherche sur cette collection statique.

3.1 Contraintes matérielles

Vous devrez vous assurer que votre système ne consomme pas plus de **1 Go de mémoire vive**, ni à l'indexation, ni à la recherche. D'autre part, votre index ne devra pas dépasser **60 % de la taille de la collection**.

Ces contraintes vous obligeront probablement à faire des choix en termes de représentation des informations, de structures de données, de fusion d'index. Tous ces choix devront bien sûr être décrits et justifiés dans le rapport.

3.2 L'indexation

Vous devrez employer **au moins deux techniques d'indexation** sur la collection : segmentation simple sans normalisation et avec *stemming*. Vous êtes libre de mettre en œuvre d'autres techniques si vous le souhaitez, et de choisir de supprimer les mots vides ou pas. Pour la pondération, la formule du **tf.idf** sera employée.

3.3 La recherche

Vous utiliserez le **modèle vectoriel** vu en cours et en TP, avec la **similarité cosinus**. Les requêtes seront de simples listes de mots-clés séparés par des espaces. Au moins deux versions du programme seront proposées : l'une s'appuyant sur l'index non normalisé, l'autre sur l'index avec *stemming*.

3.4 Évaluation

Une fois le moteur de recherche réalisé, il sera ensuite nécessaire d'évaluer sa pertinence et ses performances.

3.4.1 Pertinence

Vous évaluerez la pertinence des 20 premiers documents renvoyés par votre moteur de recherche, pour chacune des 10 requêtes ci-dessous. Vous définirez le protocole et les métriques d'évaluation vous-mêmes, avec l'aide de votre cours de recherche d'information (*"Modèles de recherche et évaluation"*, à partir de la page 48).

Les dix requêtes sont :

1. `Charlie Hedbo`
2. `volcan`
3. `playoffs NBA`
4. `accidents d'avion`
5. `laïcité`
6. `élections législatives`
7. `Sepp Blatter`
8. `budget de la défense`
9. `Galaxy S6`
10. `Kurdes`

Vous justifierez dans votre rapport le choix des métriques et la manière de juger la pertinence d'un document ; vous décrirez également les limites d'une telle méthode d'évaluation.

3.4.2 Performances

Vous fournirez toutes les mesures objectives pertinentes de la performance de votre système. Devront figurer **au minimum** :

- Le **temps de calcul** consacré à l'indexation (pour chaque index), et à la réponse moyenne à une requête.
- Vous comparerez également l'**espace disque** occupé par les différents index. Vous expliquerez les différences constatées.
- Enfin, vous estimerez la **mémoire occupée** lors de l'indexation et lors d'une requête. Vous fournirez une courbe montrant l'évolution de la consommation mémoire, en particulier pour l'indexation, par exemple avec l'aide de l'outil `jconsole`.

4 Options au choix

Vous choisirez une option (ou plusieurs si le cœur vous en dit) parmi les projets listés ci-dessous. Les enseignants sont là pour vous aider dans votre choix, ainsi que dans la progression de votre travail, en particulier pour les options les plus "aventureuses".

4.1 Option A : Extension du moteur de recherche

Technicité ★ ★ ★ ☆ ☆ Recherche ★ ☆ ☆ ☆ ☆ Exigence sur le produit fini ★ ★ ★ ★ ★

Vous sélectionnerez l'intégralité de la collection sur laquelle vous avez réalisé votre tronc commun (1,55 million de documents), et réaliserez le moteur de recherche sur cette collection, avec les mêmes contraintes matérielles, les mêmes techniques de segmentation, les mêmes stratégies de recherche, les mêmes méthodes d'évaluation.

Vous partirez du travail effectué à la première partie, puis vous augmenterez la taille de la collection progressivement (par exemple, en la multipliant par 5 à chaque essai). Un moteur de recherche qui fonctionne correctement sur "seulement" 200 000 documents sera bien mieux noté qu'un système lancé sur l'intégralité du corpus mais ne fonctionnant finalement pas.

Si vous choisissez cette option, vous êtes dispensés de l'évaluation sur le petit corpus. Vous effectuerez donc l'évaluation sur le nombre de documents que vous êtes parvenus à indexer, avec les dix requêtes suivantes :

1. `Charlie Hedbo`
2. `président du Honduras`
3. `JO de Sotchi`
4. `accidents d'avion`
5. `laïcité`
6. `élections législatives`
7. `Sepp Blatter`
8. `adoption du projet de budget 2014`
9. `contraception`
10. `Kobani Kurdistan`

4.2 Option B : Compression d'index

Technicité ★ ★ ★ ★ ☆ Recherche ★ ☆ ☆ ☆ ☆ Exigence sur le produit fini ★ ★ ★ ★ ☆

Vous sélectionnerez l'intégralité de la collection sur laquelle vous avez réalisé votre tronc commun (1,55 million de documents), et réaliserez le moteur de recherche sur cette collection.

Les contraintes matérielles : votre système ne consommera pas plus de 1 Go de mémoire vive, et votre index ne devra pas dépasser 20 % de la taille de la collection.

Vous n'utiliserez qu'une seule technique de segmentation (avec ou sans *stemming*, au choix), et vous n'évaluerez votre système que par ses performances (section 3.4.2), pas sa pertinence.

Vous partirez du travail effectué à la première partie, puis vous augmenterez la taille de la collection progressivement (par exemple, en la multipliant par 5 à chaque essai). Un moteur de recherche qui fonctionne correctement sur "seulement" 200 000 documents sera bien mieux noté qu'un système lancé sur l'intégralité du corpus mais ne fonctionnant finalement pas.

Si vous choisissez cette option, vous êtes dispensés de l'évaluation sur le petit corpus. Vous effectuerez donc l'évaluation sur le nombre de documents que vous êtes parvenus à indexer, avec les dix requêtes présentées ci-dessus pour l'option A.

4.3 Option C : Crawler

Technicité ★ ★ ★ ★ ☆ Recherche ★ ★ ☆ ☆ ☆ Exigence sur le produit fini ★ ★ ★ ☆ ☆

Vous construirez un crawler sur le modèle vu en cours : à partir de 100 URL fournies, vous collecterez 500 000 pages. Vous utiliserez le langage de programmation de votre choix.

L'évaluation de votre système se fera en termes de :

- Fonctionnement global du système
- Taux d'erreur lors du téléchargement
- Temps de calcul
- Respect des règles de politesse

Références : vous pouvez dans un premier temps vous contenter du cours sur la question, mais l'article suivant pourra vous être utile :

- Allan Heydon and Marc Najork, "Mercator : A scalable, extensible Web crawler", Conference WWW, 1999, `http://webpages.uncc.edu/sakella/courses/cloud09/papers/Mercator.pdf`

4.4 Option D : Moteur de recherche temporel

Technicité ★ ★ ★ ☆ ☆ Recherche ★ ★ ★ ☆ ☆ Exigence sur le produit fini ★ ★ ★ ☆ ☆

Vous travaillerez avec l'intégralité de la collection WEB-NEWS. La version nettoyée des données vous sera très probablement utile, et vous utiliserez la date de création du document fournie dans l'index (voir section 2).

Vous utiliserez ensuite un moteur libre comme Solr ou ElasticSearch pour indexer l'intégralité des documents, avec leur date de création. Alternativement, vous pouvez contacter votre enseignant qui mettra une instance de Solr déjà créée à votre disposition en ligne.

En vous basant sur cet index, vous construirez un moteur de recherche temporel. Ici, une requête consiste en un ou plusieurs mots-clés, accompagnée optionnellement d'une plage temporelle (date de début et de fin) sur laquelle l'utilisateur souhaite se focaliser [3]. À partir de cette requête, le système détectera les "pics de pertinence" dans le temps, c'est-à-dire les moments où les mots-clés de la requête sont particulièrement présents. Grâce à ces pics, il pourra proposer une liste de documents basés non seulement sur leur pertinence globale, mais également sur leur variété chronologique : l'utilisateur préférera des documents bien répartis dans le temps qui lui donnent une bonne idée des différents événements importants liés à sa requête, plutôt que des documents concentrés sur une petite zone temporelle.

La visualisation des résultats pourra être importante. Des outils dédiés sont présentés ci-dessous, mais vous pourrez aussi donner libre cours à votre imagination.

Des méthodologies d'évaluation de ce type de système existent dans la littérature, mais ceci n'est pas demandé ici. Tout au plus, vous pourrez proposer des protocoles d'évaluation, sans les mettre en œuvre.

- Les articles suivants sont pertinents mais pas indispensables :
  - Yang et al., "A study on Retrospective and On-Line Event Detection", in Proceedings of the annual international ACM SIGIR conference on Research and development in information retrieval, 1998, `https://www.cs.cmu.edu/~jgc/publication/A_Study_Retrospective_Online_ACM_1998.pdf`
  - Gu et al., "Detecting Hot Events from Web Search Logs", in Proceedings of the 11th international conference on Web-age information management (WAIM'10), 2010. `http://iir.ruc.edu.cn/~xuanjiang/publications/storyteller.pdf`
  - Fung et al., "Parameter free bursty events detection in text streams", in Proceedings of the 31st international conference on Very large databases (VLDB 2005), `https://www.cis.uab.edu/zhang/Spam-mining-papers/Parameter.Free.Bursty.Events.Detection.in.Text.Streams.pdf`

3. En l'absence de fenêtre temporelle spécifiée, on prendra toute la période couverte par la couverte, soit de mars 2013à septembre 2015.

  - Kessler et al., “Finding Salient Dates for Building Thematic Timelines”, in Proceedings of 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012), `http://aclweb.org/anthology-new/P/P12/P12-1077.pdf`
  - Chieu et al., “Query Based Event Extraction along a Timeline”, in Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, 2004, `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.2571&rep=rep1&type=pdf`
- Visualisation d’informations chronologiques :
  - SIMILE `http://www.simile-widgets.org/timeline/`
  - TimelineJS `http://timeline.knightlab.com/`
  - Dipity `http://www.dipity.com/`
  - ...

4.5 Option E : À la découverte du corpus

Technicité ★★☆☆☆ Recherche ★★★★★ Exigence sur le produit fini ★★☆☆☆

Cette option laisse plus libre cours à votre imagination. De nombreuses méthodes supervisées ou non supervisées permettent de manipuler un corpus et de découvrir et visualiser son contenu avec des points de vue différents.

Vous avez à votre disposition une grande quantité de contenu homogène en style (journalistique), avec des métadonnées concernant la date de création des documents, le thème, la provenance (journal). De nombreuses méthodes de traitement automatique des langues (*Natural language processing*, *NLP*) et de fouille de texte (*text mining*), parfois en lien avec les humanités numériques (*digital humanities*, *DH*) et de récentes avancées en visualisation de données (*data visualization*, *dataviz*), permettent d’étudier le corpus sous toutes les coutures.

Si vous choisissez cette option, vous proposerez une ou plusieurs stratégies visant à apporter une connaissance nouvelle sur le contenu journalistique ou sur les événements décrits dans les articles. Avant de vous lancer, vous viendrez proposer vos idées aux enseignants qui pourront estimer si vos idées sont réalisables et si elles correspondent au volume de travail attendu de vous dans ce module.

Il ne s’agira pas ici de coder des solutions à partir de rien, mais d’utiliser des outils existants pour obtenir des résultats intéressants avec notre corpus.

Vous pourrez par exemple mettre à profit les connaissances acquises dans les autres modules (en apprentissage par exemple) pour proposer des stratégies de classification, de catégorisation, d’analyse distributionnelle des mots.

Si vous avez besoin d’un moteur de recherche sur l’intégralité de la collection, vous pourrez utiliser par exemple un moteur moteur libre comme Solr ou ElasticSearch. Alternativement, vous pouvez contacter votre enseignant qui mettra une instance de Solr déjà créée à votre disposition en ligne.

Voici quelques pistes envisageables, mais toutes vos propositions seront les bienvenues :

- Comparer les différents journaux (par exemple, le Figaro et Libération) selon leur couverture de certains thèmes ou de certains événements, en termes de vocabulaire par exemple.
- Construire un classifieur thématique d’articles, en utilisant les thèmes de certains flux pour composer des ensembles d’apprentissage et de test.
- Analyser les mots, les titres ou les documents avec des méthodes récentes par réseaux de neurones (word embeddings, etc.) pour proposer une visualisation, une aide à la recherche ou d’autres outils.
- Analyser les noms de personnes dans les documents et construire un réseau (graphe) à partir des occurrences conjointes de ces noms dans les articles. Construire un outil de recherche des personnes concernées par une requête.
- Utiliser un *topic model* de type LDA pour construire des sous-ensembles thématiques, pour proposer une visualisation, une aide à la recherche ou d’autres outils.

Attention ! Ne vous lancez pas sans consulter un enseignant auparavant.

Quelques références :

- Topic models
  - Qu’est-ce que c’est ?
    - `https://www.cs.princeton.edu/~blei/papers/Blei2012.pdf`
    - `https://www.cs.princeton.edu/~blei/blei-mlss-2012.pdf`

    - http://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/
  - Implémentations :
    - L'original, en C : https://www.cs.princeton.edu/~blei/lda-c/index.html
    - en Python : https://radimrehurek.com/gensim/models/ldamodel.html
    - en Java : http://mallet.cs.umass.edu/, http://nlp.stanford.edu/software/tmt/tmt-0.4/
  - Exemples :
    - Hall et al., "Studying the History of Ideas Using Topic Models", Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008) (http://nlp.stanford.edu/pubs/hall-emnlp08.pdf)
    - "Finding key themes from free-text reviews" (http://tech.opentable.com/2015/01/12/finding-key-themes-from-free-text-reviews/)
    - Yang et al., "Topic Modeling on Historical Newspapers", Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities (2011) (http://mappingtexts.stanford.edu/whitepaper/Topic_Modeling_History.pdf)
    - Meeks et al., "The Digital Humanities Contribution to Topic Modeling", Journal of Digital Humanities (2012) (http://journalofdigitalhumanities.org/2-1/dh-contribution-to-topic-modeling/)
- Word embeddings :
  - Qu'est-ce que c'est ?
    - http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/
    - http://alexminnaar.com/word2vec-tutorial-part-i-the-skip-gram-model.html
    - http://emnlp2014.org/tutorials/8_notes.pdf
  - Implémentations de word2vec :
    - L'original, en C : https://code.google.com/p/word2vec/
    - en Python (avec un doc2vec pour modéliser les documents) : https://radimrehurek.com/gensim/models/word2vec.html
    - en Java : http://deeplearning4j.org/word2vec.html
  - Visualisation avec t-SNE (http://lvdmaaten.github.io/tsne/#examples, http://cs.stanford.edu/people/karpathy/tsnejs/)
  - Exemples :
    - Moody, "A Word is Worth a Thousand Vectors", 2015 (http://multithreaded.stitchfix.com/blog/2015/03/11/word-is-worth-a-thousand-vectors/)
- Traitement des langues et analyse politique :
  - Slapin at al., "A Scaling Model for Estimating Time-Series Party Positions from Texts", in American Journal of Political Science, 2008 (http://www.wordfish.org/uploads/1/2/9/8/12985397/slapin_proksch_ajps_2008.pdf)
  - Ploux at al., "Vers une méthode de visualisation graphique dynamique de la diachronie" (http://dico.isc.cnrs.fr/fr/diachro.html)
  - "Automatic Keyword Extraction from Presidential Speeches" http://voidpatterns.org/2015/03/automatic-keyword-extraction-from-presidential-speeches/
- Visualisation de données :
  - Data-Driven Documents (http://d3js.org/)
  - 40 JavaScript Chart and Graph Libraries http://jqueryhouse.com/javascript-chart-and-graph-libraries/
  - t-SNE (http://lvdmaaten.github.io/tsne/#examples, http://cs.stanford.edu/people/karpathy/tsnejs/)

5 Où mettre les données ?

Les quotas disponibles sur vos comptes personnels ne seront pas suffisants pour stocker vos données de travail (index, etc.). Vous pouvez utiliser l'espace partagé créé pour l'occasion :

`/projet/iri`

sur lequel vous avez des droits d'écriture. Cet espace est partagé par l'ensemble des groupes ; vous devez donc tout d'abord créer un sous-répertoire pour votre groupe, et ne travailler que dans ce sous-répertoire. Merci également de respecter les autres en ne surchargeant pas inutilement cet espace. Par exemple, merci de ne PAS faire de copie intégrale du corpus dans votre répertoire.

6 Rendu du projet

Le rendu du projet consistera en :

- Un **rapport au format pdf, ne dépassant pas 30 pages**, contenant la description de votre travail, la justification de vos différents choix et les limites identifiées. L'évaluation sera également décrite et les résultats fournis. La **répartition des tâches** entre chaque membre du projet sera également indiquée.
- Pour le travail "tronc commun" :
  - Un fichier **jar** contenant le **code java commenté intelligemment** de votre projet de tronc commun, ainsi qu'un fichier **README.txt** décrivant la marche à suivre pour utiliser le programme et toute indication utile à la compréhension du code.
  - Une archive **zip** contenant la liste des 100 premiers documents renvoyés par votre système , pour chacune des 10 requêtes proposées ci-dessus.
- Pour le travail au choix : une archive contenant l'intégralité de votre code et des données nécessaires pour l'évaluation.