Ecole Publique d'Ingénieurs en 3 ans

Rapport

PROJET 2A

Le 3 Avril 2022,
version 3.0

Arthur FESSARD,
Romaric JOLLIVET,
Dorian NAPOLI

ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

www.ensicaen.fr

Table des matières

Table des illustrations

1. Projet et son contexte

Ce projet consiste à améliorer la plateforme d'analyse forensique GDIP (Greyc Digital Investigation Platform) sur des problèmes de Natural Language Processing.

Notre projet consiste à faire de l'analyse et de la recherche de texte dans des fichiers. Nous devons ainsi pouvoir déterminer si un fichier contient un ou plusieurs mots de façon exacte, trouver des expressions régulières plus ou moins complexes, déterminer la langue utilisée dans le fichier, identifier des entités textuelles telles que des dates et analyser le sentiment dégagé par le texte.

La plateforme GDIP a pour but d'être utilisée de manière professionnelle pour mener des enquêtes forensiques. Elle pourrait potentiellement être utilisée par la gendarmerie pour rapidement analyser un disque dur rempli de fichiers afin de déterminer automatiquement quels sont les fichiers les plus intéressants.

Nous détaillons dans ce rapport notre démarche afin de mener ce projet à bien.

2. Objectifs du Projet

Ce projet est découpé en deux objectifs généraux.

Le premier est d'analyser et de mesurer les performances d'outils de recherche et d'analyse de texte. On pourra ainsi qualifier la performance de ces outils et ne retenir que les meilleurs.

Le second est de développer un filtre pour la plateforme GDIP (Greyc Digital Investigation Platform) qui implémente ces outils. Cela consiste donc en une intégration sur la base de code existante.

Nous avons identifié différentes tâches se rattachant à ces objectifs et permettant de les réaliser de manière concrète.

Nous devons tout d'abord créer un corpus de fichiers test, en y mettant différents types de fichiers (txt, pdf, etc…), avec différents contenus (mail, rapport, notes, etc…), différentes langues (anglais, français, etc…) et différents sentiments (joie, colère, etc…). Ce corpus de fichiers sera utilisé pour qualifier la performance des différents outils.

Nous devons en parallèle réaliser un état de l'art des ressources utiles pour le projet. Cela peut être des outils, mais aussi des techniques, méthodes ou papiers scientifiques pouvant nous aider à accomplir ce projet.

Il nous faudra ensuite tester les outils précédemment trouvés sur notre corpus de fichiers afin d'évaluer leur performance. Le développement de scripts pour automatiser ces tests est donc nécessaire.

Une autre tâche, quant à elle optionnelle, consiste à ajouter des fonctionnalités aux outils trouvés et/ou les fusionner pour tirer parti de leurs avantages respectifs et ainsi obtenir de meilleures performances.

Enfin, il nous faudra intégrer les outils que nous avons retenu dans un filtre utilisable pour la plateforme GDIP.

3. Méthodologie utilisée, répartition du travail

Afin de réaliser ce projet de manière efficace, nous nous sommes réparti le travail de manière équitable.

Durant le premier semestre, nous nous étions plutôt penchés sur la construction de corpus de texte et la recherche d'outils de recherche et d'analyse de texte. Nous nous étions concentrés sur deux fonctionnalités : la détection de langue et de sentiment. Pour cela nous avions décidé de couper notre groupe en deux, une personne s'occupait de créer les corpus, tandis que les deux autres commençaient la recherche d'outils.

Pendant le second semestre, nous avons poursuivi la recherche et les tests de ces outils. En parallèle, nous avons décidé de tester deux nouvelles fonctionnalités : la recherche de texte et l'extraction de dates dans des fichiers.

Pour cette recherche, nous nous sommes séparé le travail pour que chacun travaille sur une fonctionnalité : le premier sur l'analyse de sentiments, le deuxième sur la détection de langue et enfin le dernier sur la recherche d'entités textuelles.

Nous avons pour cela utilisé une méthodologie Agile en fonctionnant par incrémentation et en augmentant au fur et à mesure les outils que nous testons. Nous avons pu effectuer plus de réunions avec notre client par rapport au premier semestre, il était de ce fait plus facile d'avoir un retour et savoir sur quoi se concentrer.

4. Outils utilisés

Pour faciliter la collaboration et l'organisation du code, nous avons utilisé le logiciel de gestion de versions Git. Cela nous a permis de travailler sur la même base de code et de facilement mettre en commun nos travaux respectifs. Nous avons ensuite utilisé l'application GitLab afin de mettre en ligne notre dépôt git.

Afin de trouver plusieurs outils liés à la recherche et l'analyse de texte, nous avons beaucoup utilisé les sites d'hébergement de code Github et GitLab.

Figure 1 Logo de GitLab

Pour réaliser notre corpus, des sites tels que Kaggle nous ont permis d'exploiter des corpus de fichiers déjà existants et classifiés. Nous avons également pu exploiter des corpus provenant d'état de l'art fait au préalable dans des papiers scientifiques.

Figure 2 Logo de Kaggle

5. Travail réalisé, objectifs atteints

Nous avions réussi à poser des bases solides sur notre projet au premier semestre , nous avons donc pu poursuivre dans cette lancée pour ce deuxième semestre.

Comme dit précédemment au premier semestre, nous nous étions concentrés sur deux fonctionnalités : l'analyse de sentiments et la détection de langue.

Pour le deuxième semestre, nous avons ajouté une fonctionnalité qui est la recherche d'entité textuelle dans les fichiers et avons poursuivi le travail de recherche et de test sur les deux autres fonctionnalités. Enfin nous avons pu implémenter les filtres pour la plateforme GDIP correspondant à ces fonctionnalités.

5.1. Construction du corpus

Nous avons débuté avec la création d'un corpus de texte, qui à cette date contient une centaine de fichiers en .txt et une centaine de fichiers en .pdf. Nous avons diversifié les contenus de ces fichiers en y mettant des fichiers en français et en anglais, des mails, des rapports, ou encore des listes d'objets.

Afin de tester chaque fonctionnalité, nous avons créé un corpus par fonctionnalité que nous voulions tester. Pour ce faire nous nous sommes aidés de corpus déjà créés trouvés sur internet, comme sur Kaggle par exemple.

Pour le corpus de détection de langues, de nombreux corpus sont disponibles sur internet. Nous en avons sélectionné un contenant les 48 langues les plus connues et environ 8 Millions de phrases (10, s.d.). Chaque phrase fait environ 35 caractères de long et contient environ 6 mots. Il s'agit d'un fichier au format CSV et pour chacune de ces phrases, la langue correspondante est associée. Nous avons dû effectuer quelques modifications sur les noms des langues associés afin qu'ils correspondent au résultat renvoyé par les outils (standard ISO 639-1).

Pour le corpus d'analyse de sentiments, nous avions initialement décidé d'utiliser le polarity dataset v2.0 (11, s.d.), un corpus de 2000 fichiers textes correspondant à des critiques de films classifiés par sentiments, avant d'en choisir un plus diversifié pour une meilleure fiabilité sur nos résultats. Les tests d'analyse de sentiments finaux ont donc été réalisés à partir d'un corpus équitablement divisé entre sentiment positif et négatif constitué de 1000 critiques Amazon (environ 55 caractères de long, 10 mots), 1000 critiques de films (environ 83 caractères, 15 mots) et 1000 critiques culinaires issues de Yelp (environ 58 caractères, 11 mots). Pour vérifier la robustesse de notre choix final nous testerons également le filtre choisi sur 10000 tweets équitablement répartis du corpus Sentiment140 (environs 61 caractères, 12 mots), ce dernier étant habituellement plus difficile à prédire que de simples avis.

Nous nous sommes focalisés majoritairement sur des corpus assez légers en taille afin de garder un temps raisonnable lors de l'exécution de nos outils. Nos machines n'étant pas très puissantes, il nous a paru plus intéressant de favoriser une exécution rapide afin de pouvoir tester rapidement après chaque modification de nos scripts de tests.

Nous avons donc créé les 4 corpus ci-dessous :

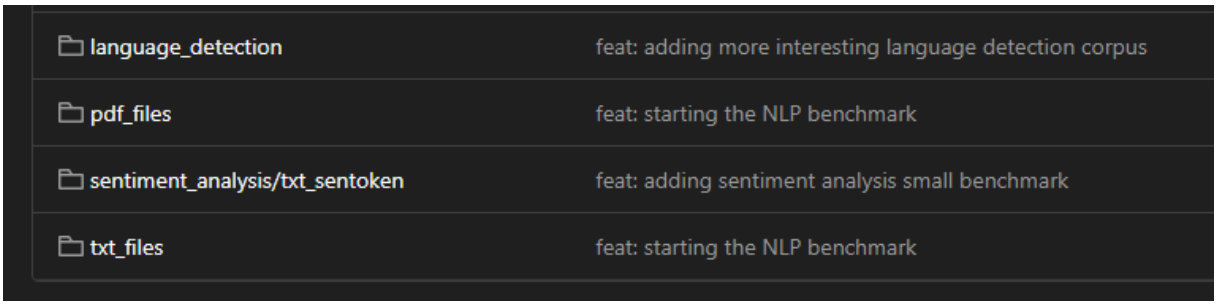

Figure 3 Corpus créés

Nous avons en parallèle essayé de lister de manière exhaustive les différents outils d'analyse et de recherche de texte.

5.2. Méthodologie d'évaluation

Afin d'évaluer les différents outils, nous avons défini un protocole d'évaluation, que nous avons amélioré au fil de ce projet.

Nous exécutons chaque outil relatif à une fonctionnalité spécifique (analyse de sentiment, détection de langue, ...) sur un même corpus. Nos corpus étant déjà classifiés, nous pouvons déterminer si cet outil renvoie la bonne classification pour chaque fichier. Lorsque tous les fichiers du corpus ont été examinés, nous pouvons calculer la métrique de précision de classification de chacun de ces outils en divisant le nombre de bons résultats par le nombre de fichiers dans le corpus. Cette métrique d'évaluation n'est pas utile dans le cadre de la recherche de texte dans les fichiers.

Pour le premier semestre, il s'agissait de la seule métrique d'évaluation que nous considérions.

Nous avons pu ajouter une métrique de mesure de temps d'exécution durant ce second semestre afin de déterminer les outils les plus performants en termes de vitesse. Cette métrique de temps d'exécution est très intéressante pour ce projet.

En effet, la plateforme GDIP sera utilisée pour scanner plusieurs giga-octets voire plusieurs téraoctets de fichiers de manière automatisée, il est donc important que les outils que nous évaluons présentent un bon compromis entre précision de classification et vitesse d'exécution.

5.3. Travail sur les outils d'analyse de sentiments

La première fonctionnalité que nous avons explorée est l'analyse de sentiments. L'analyse de sentiments permet de déterminer quel est le sentiment général dégagé par un texte. En pratique cela s'est traduit par une vérification binaire du sentiment d'un texte, à savoir positif ou négatif.

Pour cela, nous avons sélectionné trois outils qui nous semblait intéressants et sur lequel nous avons concentré nos recherches.

Ces trois outils sont les suivants :

- NLTK (1)
    - Il s'agit d'une suite de bibliothèques logicielles et de programmes de traitement naturel du langage
    - Conçue pour le traitement naturel symbolique et statistique du langage anglais.
    - Rassemblement des algos les plus communs du traitement naturel du langage. Nous utiliserons son modèle pré entrainé "VADER", particulièrement adapté aux phrases courtes et aux abréviations ce qui est caractéristique des réseaux sociaux.
- SpaCy (2)
    - Il s'agit d'une des principale bibliothèque du langage python pour le traitement naturel du langage
    - C'est une bibliothèque de bas niveau mais intuitive et performante
    - Il est possible de traiter et de comprendre de larges volumes de texte
- TextBlob (4)
    - Il s'agit d'une bibliothèque du langage python pour le traitement naturel du langage
    - Très simple d'utilisation, il s'agit d'une version simplifiée de SpaCY
    - TextBlob repose sur les bibliothèques NLTK et Pattern en combinant les deux

Nous avons utilisé la bibliothèque Panda de python afin de traduire en datagramme le corpus de critiques ainsi que "Sentiment140" (9, s.d.) pour le test de robustesse, tous deux sont initialement classifiés en textes positifs ou négatifs et sont issues du site communautaire Kaggle.

Nous avons commencé nos tests sur la version pré entrainé de ces librairies lorsqu'elles existaient, ces dernières étant préférables à nos yeux pour leur facilité d'utilisation et leur polyvalence, puisque déjà entrainé sur une variété de scénario par leurs créateurs. Ainsi nous avons utilisé le modèle pré-entrainé VADER de NLTK, le modèle communautaire Spacy "eng_spacysentiment" et le modèle par défaut de Textblob.

Nous nous sommes vite aperçus que le modèle pré entrainé de Spacy était insatisfaisant de par sa lenteur (à savoir 792 secondes pour notre corpus de 3000 critiques) et nous nous sommes donc tourné vers un modèle que nous avons-nous mêmes entrainé dans l'espoir d'obtenir des performances plus rapides. Le paramètre d'entrainement que nous avons déterminé comme optimal pour notre modèle Spacy a été un échantillon de test de taille 0.25 soit un quart du corpus.

Nous avons également testé différents tokenizer pour nos bibliothèques lorsqu'elles nous en laissaient la possibilité. Si nous avons obtenu de meilleurs résultats avec le tokenizer par défaut de la bibliothèque pour NLTK, nous avons pu améliorer les résultats de notre modèle Spacy en utilisant plutôt un tokenizer maison qui sépare et normalise les mots en minuscule tout en supprimant les "stop words".

Bien que nous ayons pu sensiblement améliorer la vitesse d'exécution de Spacy et, dans une mesure plus légère, la précision de ses prédictions, notre modèle est resté en dessous de ses concurrents.

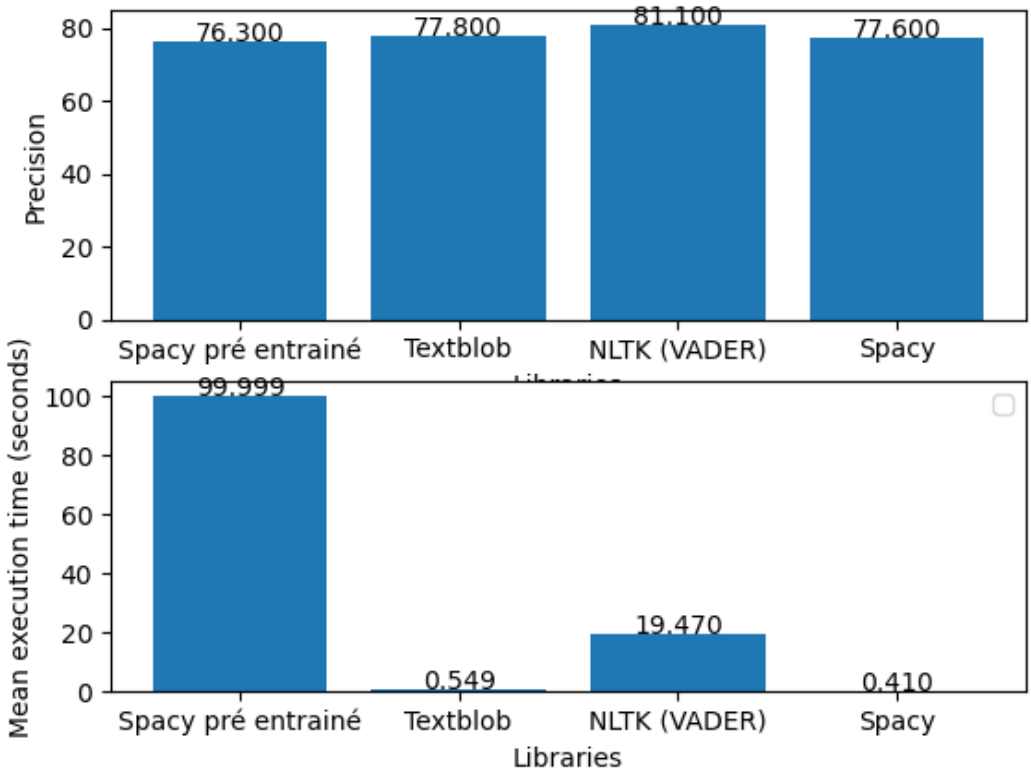

Figure 4 Taux de précision et vitesse d'exécution pour les outils d'analyse de sentiment sur le corpus de 3000 critiques

Nous avons donc finalement conclu de nos tests sur le corpus de critiques que Textblob et NLTK étaient les modèles les plus performants. Si l'idée de combiner les deux filtres nous a semblé intéressante, cette dernière n'a pas portée ses fruits puisque nous obtenions constamment des résultats inférieurs à ceux de NLTK dont un maximum à 80.14% de précision pour une exécution de 20 secondes.

Comme prévu précédèment, nous avons également effectué un test de robustesse de nos deux outils choisis sur le corpus de 10000 tweets, plus difficile à prédire, qui a confirmé nos observations: le modèle VADER de NLTK a l'avantage de la précision alors que Textblob brille par ses résultats plus rapide tout en restant satisfaisant en comparaison.

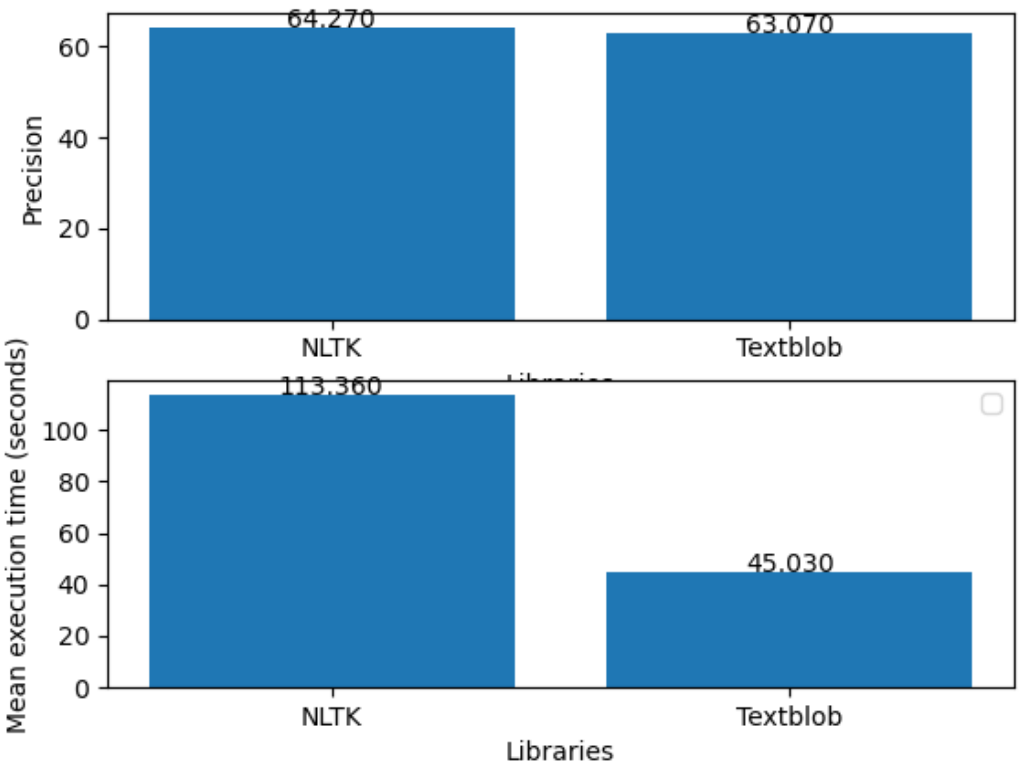

Figure 5 Taux de précision et vitesse d'exécution pour les outils d'analyse de sentiment retenus sur le corpus de 10000 tweets

5.4. Travail sur les outils de détection de langue

La deuxième fonctionnalité que nous avons explorée est la détection de langue. Cela nous permet de déterminer quelle est la langue majoritaire d'un fichier. Nous avons durant ce second semestre pu tester davantage d'outils de détection de langue.

Nous avons identifié les bibliothèques python de détection de langue suivantes :

- Langdetect (5)
    - Portage de la bibliothèque language-detection de Google en python
- Py3Langid (7)
    - Il s'agit d'une version améliorée de Langid (6) en termes de vitesse d'exécution
- gcld3 (18, s.d.)
    - Google Compact Language Detector v3

- Lingua (14, s.d.)
- XLM-roberta (15, s.d.)
    - Modèle pré-entrainé disponible sur le site Hugging Face.
- pyfranc (17, s.d.)
- Fasttext (16, s.d.)
    - Il s'agit d'une bibliothèque de NLP créé par Facebook AI.

Ces bibliothèques sont pour la plupart basées sur des modèles pré-entrainés. La détection de langue est un problème qui existe depuis longtemps et pour lequel nous avons de très bons résultats. Nous pouvons observer des résultats intéressants en fonction des différents outils.

Nous avons décidé d'omettre les librairies XLM-roberta et pyfranc car elles étaient très lentes en termes de vitesse d'exécution et peu précises en termes de détection de langues.

Il est important de noter que les résultats sont variables en fonction des corpus de test utilisés. Certaines bibliothèques sont plus performantes sur certaines langues, et certaines supportent plus de langues que d'autres. Pour nos tests, nous nous sommes assurés que tous les outils supportaient toutes les langues du corpus pour que l'évaluation soit juste.

Voici les résultats que nous avons obtenu sur le corpus de grande taille évoqué précédemment (10, s.d.) (~ 8M phrases, 48 langues, ~500 méga-octets) :

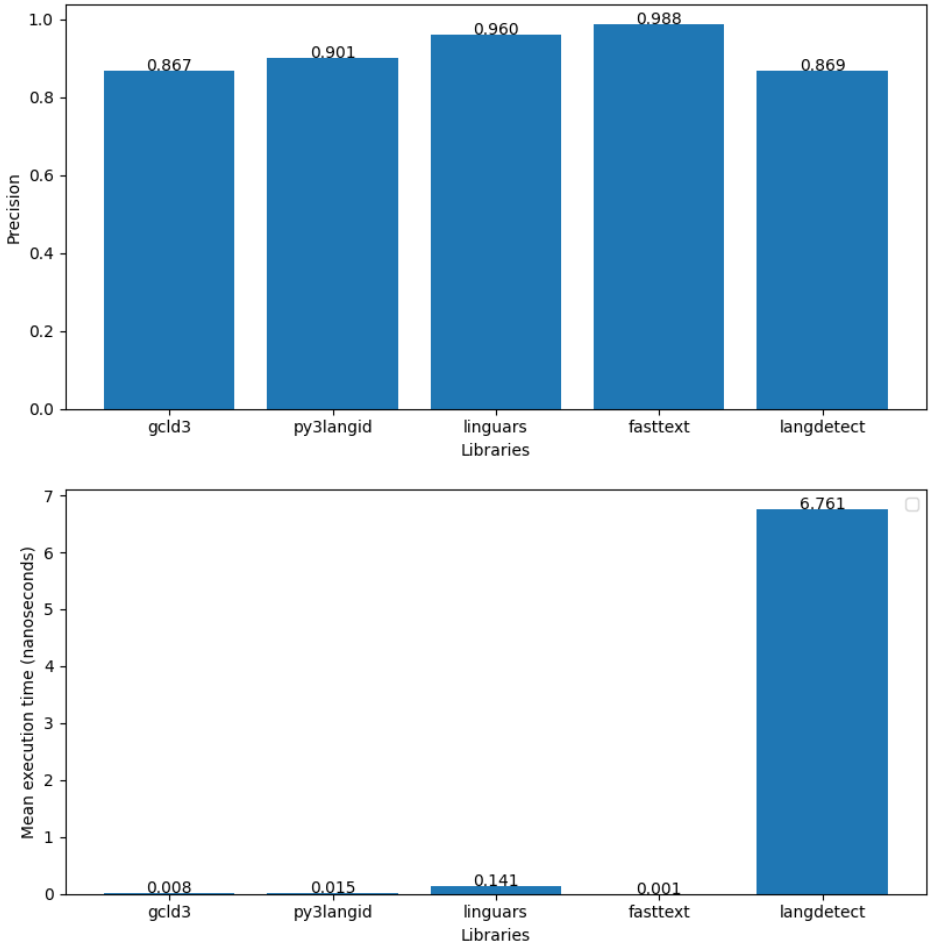

Figure 6 Histogramme montrant le taux de précision et la vitesse d'exécution pour les outils de détection de langue

On note donc que la bibliothèque Fasttext (16, s.d.) est la plus performante en termes de précision sur ce corpus ainsi qu'en termes de vitesse d'exécution, c'est donc assez impressionnant.

Au vu de ces résultats nous avons décidé de conserver les bibliothèques suivantes pour notre filtre GDIP : fasttext, gcld3, py3langid et lingua.

Nous avons abandonné langdetect car il n'est pas particulièrement précis et très lent par rapport aux autres outils.

5.5. Travail sur les outils de recherche de texte

La troisième fonctionnalité que nous avons explorée est la recherche de texte dans les documents. Cela nous permet de déterminer si un mot ou une phrase est présent dans un texte.

Nous avons identifié et testé les bibliothèques python suivantes :

- NLTK (1)
- grep (13, s.d.)
- ripgrep (12, s.d.)
- SpaCy (2)
- Regex natif de python avec le module re

Contrairement aux trois autres fonctionnalités, la recherche de texte dans des fichiers est déjà totalement implémentée. Il n'était donc ici pas utile de tester la précision des outils. Nous avons donc testé nos outils seulement sur leur vitesse étant donné qu'en termes de précision ils sont tous égaux.

Nous avons effectué des tests sur un bon nombre de corpus et sur différents mots ou phrases, avec à chaque fois le même ordre des outils concernant la vitesse.

Voici les résultats que nous avons obtenu sur le corpus de critiques de films positives (11, s.d.), en cherchant le mot « joke » 50 fois dans ce corpus et en prenant la moyenne des temps :

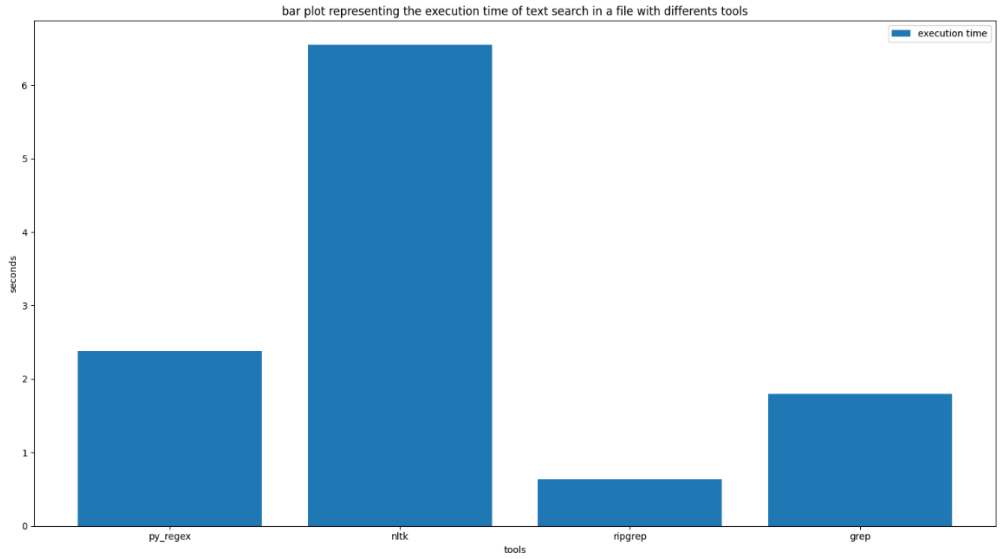

Figure 7 Histogramme montrant la vitesse de recherche de texte de différents outils

On note ainsi que la bibliothèque ripgrep (12, s.d.) est la plus performante pour effectuer ce genre de recherche, c'est donc avec celle-ci que nous avons décidé de créer le filtre qui devra être implémenté à la plateforme GDIP.

5.6. Extraction de dates

La dernière fonctionnalité que nous avons explorée est l'extraction de dates. Cela consiste à extraire les dates sous toute forme dans un fichier.

Nous avons identifié et testé les bibliothèques python suivantes :

- Datefinder (19, s.d.)
- Date-detector (20, s.d.)
- Date-extractor (21, s.d.)
- Dateparser (22, s.d.)
- Goose3 (23, s.d.)
- Extract-date (24, s.d.)
- Htmldate (25, s.d.)

Ces bibliothèques fonctionnent pour la plupart sur des systèmes d'expressions régulières et de modèle pré-entrainé.

Nous avons décidé de ne pas retenir les bibliothèques goose3 et htmldate car elles ne renvoient qu'une seule date pour tout le fichier, ce que nous n'avons pas estimé intéressant.

Voici les résultats que nous avons obtenus sur un corpus de 500 fichiers html de taille moyenne (environ 40000 caractères) :

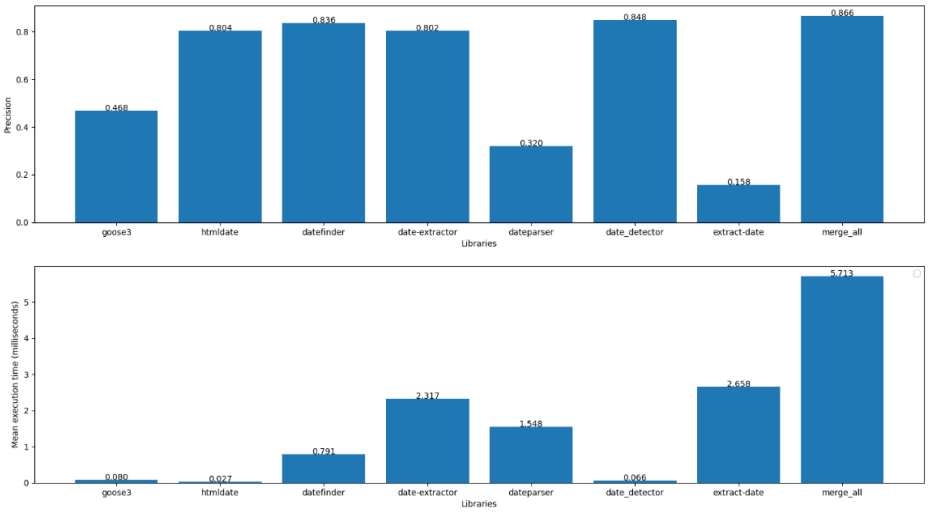

Figure 8 histogramme montrant le taux de précision et la vitesse d'exécution pour les outils d'extraction de date

Suite à ces résultats, nous avons décidé de retenir les bibliothèques Datefinder, Date-detector et Date-extractor pour le filtre GDIP.

5.7. Filtres

Une fois les meilleurs outils identifiés nous avons pu implémenter les filtres pour la plateforme GDIP pour certaines fonctionnalités. Cette implémentation n'a pas été une étape compliquée car nous savions déjà comment bien utiliser nos outils, il nous a juste fallu les adapter à la base de code de la plateforme GDIP.

Nous avons inclus dans notre filtre d'analyse de sentiment un compromis entre les avantages des deux librairies retenues à savoir 2 modes: un mode "rapide" utilisant Textblob et un mode "précision", par défaut, exploitant NLTK.

Pour le filtre de détection de langue et d'extraction de dates, une stratégie de fusion des résultats des différents outils a été utilisé. Cette stratégie est simple : pour la détection de langue on renvoie la langue majoritairement choisie et pour les dates, on combine toutes les

dates trouvées en supprimant les doublons. Cela permet de combiner les forces des différents outils bien que cela augmente le temps d'exécution.

Pour le filtre de recherche de texte, nous n'avons pas utilisé de fusion de résultat car cela n'était pas nécessaire en l'absence d'attente de précision puisque les mots sont toujours identifiés correctement.

6. Difficultés rencontrées

Nous avons rencontré diverses difficultés lors du développement de ce projet.

La première fut l'organisation du travail en équipe. En effet, le projet étant vaste et difficile à appréhender au premier abord, nous avons mis du temps à bien séparer les tâches de chacun.

Trouver des corpus pertinents pour nos tests a demandé un peu de réflexion et ces derniers ont été en constante évolution. Nous avons donc dû refaire nos tests à chaque changement dans un corpus.

La prise en main des différentes bibliothèques pythons que nous avons utilisés a aussi été assez difficile. Nous avons eu beaucoup de problèmes de version non compatibles ou dépréciées, etc.

Enfin, un des membres de notre équipe a arrêté son parcours à l'ENSICAEN en plein milieu de ce projet. Cela a donc perturbé toute notre organisation en plus d'impacter notre efficacité. Nous avons donc dû revoir la répartition du travail, et travailler chacun un peu plus afin d'atteindre les mêmes objectifs.

7. Ouverture sur une possible suite

La plateforme GDIP est en constante évolution, et nous avons pu y apporter notre modeste contribution grâce à ces quelques filtres. Nous n'avons malheureusement pas eu le temps de les tester en situation réelle, ce qui nous semblait pourtant intéressant.

Par ailleurs, hormis pour la détection de langues, nous nous sommes contentés d'effectuer nos tests sur des corpus anglophones car c'est le langage pour lequel est conçu la majorité des outils. Il serait alors pertinent d'essayer d'ouvrir nos filtres à d'autres langues.

Nous avons hâte de voir ce à quoi ressemblera la plateforme GDIP dans le futur. Nous espérons que de futurs étudiants contribueront et implémenteront de nouveaux filtres.

8. Conclusion

Ce projet nous a permis dans un premier temps d'améliorer notre aptitude à travailler en équipe. En effet, il a fallu que nous réussissions à nous partager efficacement le travail pour pouvoir avancer le plus efficacement possible, tout en s'entraidant lorsque la compréhension nous faisait défaut.

Il nous a aussi permis de nous entraîner au développement en langage python et de nous familiariser avec plusieurs bibliothèques. Nous avons pu apprendre à réaliser un état de l'art sur un domaine spécifique, ici le Natural Language Processing. Nous avons également acquis une méthodologie d'évaluation afin de tester différents outils, ce qui nous sera sans aucun doute utile par la suite.

Ce projet est motivant car nous savons que notre travail peut avoir une réelle utilité. En effet, la plateforme d'analyse forensique GDIP que nous essayons d'améliorer a vocation à servir dans des secteurs tels que les enquêtes policières, et apporter sa pierre à l'édifice est exaltant.

Ce projet a donc déjà été très enrichissant du début à la fin.

9. Références

1. (s.d.). Récupéré sur https://github.com/nltk/nltk

10. (s.d.). Récupéré sur https://tatoeba.org/en/downloads

11. (s.d.). Récupéré sur https://www.cs.cornell.edu/people/pabo/movie-review-data/

12. (s.d.). Récupéré sur https://github.com/BurntSushi/ripgrep

13. (s.d.). Récupéré sur https://en.wikipedia.org/wiki/Grep

14. (s.d.). Récupéré sur https://github.com/pemistahl/lingua

15. (s.d.). Récupéré sur https://huggingface.co/xlm-roberta-large

16. (s.d.). Récupéré sur https://fasttext.cc/docs/en/language-identification.html

17. (s.d.). Récupéré sur https://github.com/cyb3rk0tik/pyfranc

18. (s.d.). Récupéré sur https://github.com/google/cld3

19. (s.d.). Récupéré sur https://github.com/akoumjian/datefinder

2. (s.d.). Récupéré sur https://github.com/explosion/spaCy

20. (s.d.). Récupéré sur https://github.com/ishirav/date-detector

21. (s.d.). Récupéré sur https://github.com/DanielJDufour/date-extractor

22. (s.d.). Récupéré sur https://github.com/scrapinghub/dateparser/

23. (s.d.). Récupéré sur https://github.com/goose3/goose3

24. (s.d.). Récupéré sur https://github.com/gajus/extract-date

25. (s.d.). Récupéré sur https://github.com/adbar/htmldate

3. (s.d.). Récupéré sur https://github.com/RaRe-Technologies/gensim

4. (s.d.). Récupéré sur https://github.com/sloria/TextBlob

5. (s.d.). Récupéré sur https://github.com/Mimino666/langdetect

6. (s.d.). Récupéré sur https://github.com/saffsd/langid.py

7. (s.d.). Récupéré sur https://github.com/adbar/py3langid

8. (s.d.). Récupéré sur https://medium.com/district-data-labs/modern-methods-for-sentiment-analysis-694eaf725244

9. (s.d.). Récupéré sur http://help.sentiment140.com/for-students/

ENSI CAEN

ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Merci

G'DIP

Ecole Publique d'Ingénieurs en 3 ans

6 boulevard Maréchal Juin, CS 45053
14050 CAEN cedex 04

ministère éducation nationale enseignement supérieur recherche

Normandie Université

Cti Commission des titres d'ingénieur

cdefi Conférence des Directeurs des Écoles Françaises d'Ingénieurs

European Accreditation of Engineering Programmes EUR-ACE®

CONFÉRENCE DES GRANDES ÉCOLES