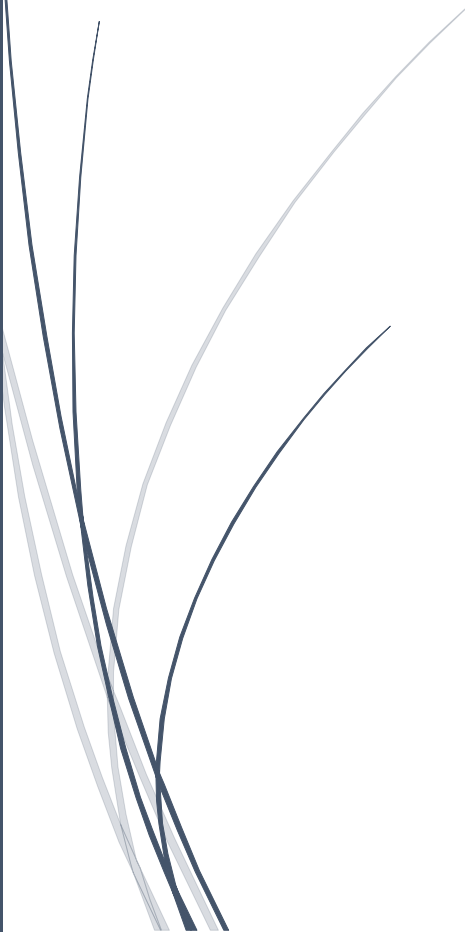
A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

18/07/2024

Projet d'heuristique et algorithmique de texte

Membres

Several thin, curved lines in dark blue and light grey originate from the bottom left and curve upwards and to the right.

BADO Inès
BAMBORE Eudes Arnaud
MIEN Kounabé Paulin
SAWADOGO Cyntia Nadine
YAMEOGO Angelo M. J. K. 1er Jumeau

Table des matières

1. Introduction.....	1
2. Algorithme Naïf de Brute-force	1
Description de l'algorithme	1
3. Avantages et inconvénients de l'algorithme	3
4. Recherche et compréhension des concepts théoriques	3
5. Comparaison avec d'autres algorithmes de recherche de motifs.....	4
6. Développement de l'Application	4
7. Présentation de l'application	5
8. Résumé des objectifs atteints.	6
9. Conclusion	7
10. Annexes	Erreur ! Signet non défini.

1. Introduction

L'analyse de similarité entre documents est un domaine important dans le traitement automatique du langage naturel, ayant des applications variées telles que la détection de plagiat, la comparaison de textes juridiques, et l'analyse de documents scientifiques. Ce projet a pour objectif de développer une application capable d'analyser la similarité entre deux documents en utilisant l'algorithme naïf de Brute-force. Cet algorithme, bien que simple, permet de rechercher des motifs communs entre deux ensembles de données textuelles de manière exhaustive. L'application que nous proposons permet de charger deux fichiers texte, d'identifier les occurrences de phrases ou de mots similaires, et de fournir des statistiques détaillées sur le niveau de similarité entre les documents. Les principaux objectifs de ce projet sont de comprendre et d'implémenter l'algorithme naïf de Brute-force, d'analyser des documents pour trouver des motifs communs, de calculer des métriques de similarité, et de créer une interface utilisateur intuitive pour charger les documents et afficher les résultats. Ce rapport décrit en détail les étapes suivies pour atteindre ces objectifs, depuis la recherche et l'implémentation de l'algorithme jusqu'au développement de l'application, en passant par l'analyse des résultats et les tests de validation effectués.

2. Algorithme Naïf de Brute-force

Description de l'algorithme

❖ Fonctionnement général de l'algorithme

L'algorithme naïf de Brute-force, aussi appelé algorithme de recherche exhaustive, est une méthode simple mais efficace pour trouver des motifs communs entre deux ensembles de données textuelles. Le principe de base de cet algorithme consiste à comparer chaque sous-ensemble de caractères du premier document avec chaque sous-ensemble de caractères du second document. Voici les étapes générales de l'algorithme :

❖ Initialisation

L'algorithme commence par prendre deux documents texte en entrée. Ces documents peuvent être de tailles différentes et contiennent le texte à comparer pour trouver des similarités.

```
document1 = "texte du premier document"
```

```
document2 = "texte du second document"
```

Les documents sont stockés sous forme de chaînes de caractères, et l'algorithme se prépare à les analyser en les parcourant de manière exhaustive.

❖ Comparaison des sous-chaînes

Pour chaque position i dans le premier document, l'algorithme extrait une sous-chaîne de longueur variable et la compare avec toutes les sous-chaînes de même longueur dans le second document. Voici comment cela se passe :

Extraction des sous-chaînes : Pour une position donnée i dans le premier document, l'algorithme extrait une sous-chaîne de longueur l commençant à cette position.

```
for i in range(len(document1)):
    for l in range(1, len(document1) - i + 1):
        sous_chaine1 = document1[i:i+l]
```

Comparaison avec le second document : Pour chaque sous-chaîne extraite du premier document, l'algorithme la compare avec toutes les sous-chaînes de même longueur dans le second document.

```
for j in range(len(document2) - l + 1):
    sous_chaine2 = document2[j:j+l]
    if sous_chaine1 == sous_chaine2:
        # Correspondance trouvée
```

❖ Identification des motifs

Lorsqu'une correspondance est trouvée entre une sous-chaîne du premier document et une sous-chaîne du second document, cette correspondance est enregistrée. Cela peut se faire en stockant les positions ou les sous-chaînes correspondantes.

```
correspondances = []
if sous_chaine1 == sous_chaine2:
    correspondances.append((i, j, l))
```

Cette étape permet de garder une trace de toutes les similarités trouvées entre les deux documents.

Itération

Le processus se répète pour toutes les positions i dans le premier document et toutes les longueurs l possibles des sous-chaînes. Cela signifie que l'algorithme va explorer toutes les combinaisons possibles pour s'assurer qu'aucun motif commun n'est manqué.

```
for i in range(len(document1)):
    for l in range(1, len(document1) - i + 1):
        sous_chaine1 = document1[i:i+l]
```

```
for j in range(len(document2) - l + 1):  
    sous_chaine2 = document2[j:j+l]  
    if sous_chaine1 == sous_chaine2:  
        correspondances.append((i, j, l))
```

Cette itération exhaustive garantit que l'algorithme identifie toutes les sous-chaînes communes entre les deux documents.

3. Avantages et inconvénients de l'algorithme

❖ **Avantages :**

- Simplicité : L'algorithme est facile à comprendre et à implémenter.
- Complétude : En comparant toutes les sous-chaînes possibles, il garantit de trouver tous les motifs communs entre les deux documents.

❖ **Inconvénients :**

- Complexité temporelle : L'algorithme a une complexité temporelle élevée, en $O(n*m)$, où n et m sont les tailles des deux documents. Cela le rend inefficace pour les grands volumes de données.
- Performance : En raison de sa nature exhaustive, l'algorithme peut être très lent et consommateur de ressources pour des documents volumineux.

4. Recherche et compréhension des concepts théoriques

Pour bien comprendre l'algorithme naïf de Brute-force, il est essentiel de se plonger dans les concepts de base de la recherche de motifs dans les textes. La recherche de motifs consiste à identifier des sous-chaînes (ou motifs) dans une chaîne de texte donnée. Cette opération est fondamentale en informatique, notamment dans le domaine de la bioinformatique pour la comparaison de séquences génétiques, dans la détection de plagiat, ou encore dans le traitement de texte.

❖ **Principe de la force brute**

L'algorithme naïf de Brute-force repose sur le principe de la force brute. La force brute est une méthode de résolution de problème qui consiste à essayer toutes les combinaisons possibles pour trouver une solution. En d'autres termes, l'algorithme compare chaque sous-chaîne du premier document avec chaque sous-chaîne de même longueur du second document, afin de ne manquer aucune correspondance.

❖ **Complexité temporelle**

L'algorithme naïf de Brute-force a une complexité temporelle de $O(n*m)$, où n est la longueur du premier document et m la longueur du second document. Cela signifie que le temps d'exécution de l'algorithme augmente de manière exponentielle avec la taille des documents. Pour des documents de grande taille, cela peut rendre l'algorithme très inefficace. Cependant, pour des documents de taille modeste, cette complexité peut être acceptable.

5. Comparaison avec d'autres algorithmes de recherche de motifs

Il existe plusieurs autres algorithmes de recherche de motifs qui sont plus optimisés que l'algorithme naïf de Brute-force. En voici quelques-uns :

- ❖ **Algorithme de Knuth-Morris-Pratt (KMP)** : Cet algorithme améliore l'efficacité en prétraitant le motif pour déterminer où les prochaines comparaisons doivent se produire, évitant ainsi les comparaisons inutiles. La complexité temporelle est $O(n + m)$.
- ❖ **Algorithme de Boyer-Moore** : Utilise deux heuristiques pour sauter des sections du texte, rendant la recherche beaucoup plus rapide dans la pratique. Il a une complexité temporelle moyenne de $O(n/m)$.
- ❖ **Algorithme de Rabin-Karp** : Utilise le hachage pour comparer des sous-chaînes. Bien que sa complexité temporelle moyenne soit $O(n + m)$, il peut être inefficace en cas de nombreuses collisions de hachage.
- ❖ **Algorithme de suffixe (Suffix Array et Suffix Tree)** : Ces algorithmes utilisent des structures de données avancées pour accélérer la recherche de motifs et ont des complexités temporelles linéaires dans certains cas, mais sont plus complexes à implémenter.

6. Développement de l'Application

Nous avons choisi d'utiliser le framework Django pour créer une interface utilisateur conviviale. Django a été sélectionné en raison de ses robustes fonctionnalités de gestion de fichiers et de création d'interfaces web, qui nous ont permis de développer rapidement une application performante et intuitive. En utilisant Django, nous avons pu bénéficier de ses nombreux modules intégrés, simplifiant ainsi le développement et la maintenance de notre application.

❖ Développement de l'interface utilisateur :

Pour offrir une expérience utilisateur agréable et efficace, nous avons développé une interface accueillante. Cette interface permet aux utilisateurs de charger deux fichiers texte pour comparaison. Voici quelques-unes des fonctionnalités clés que nous avons implémentées :

❖ Téléchargement de fichiers :

L'interface utilisateur permet aux utilisateurs de sélectionner et de téléverser facilement les fichiers à comparer. Un formulaire de téléchargement intuitif guide les utilisateurs tout au long du processus, assurant ainsi une expérience sans heurts.

❖ Retour visuel :

Une fois les fichiers téléversés, l'interface fournit un retour visuel clair, indiquant le statut des fichiers téléchargés et permettant de vérifier qu'ils ont été correctement chargés.

❖ Navigation simplifiée :

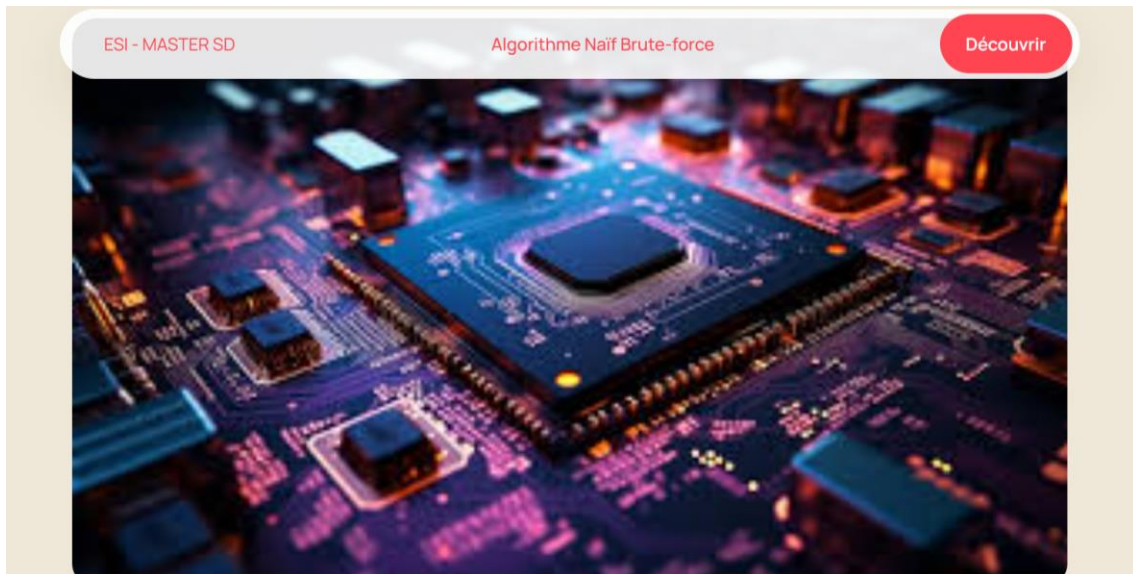
Nous avons conçu une interface épurée et facile à naviguer, permettant aux utilisateurs de se concentrer sur l'objectif principal de l'application sans être distraits par des éléments superflus.

❖ Accessibilité et réactivité :

L'interface est conçue pour être accessible sur différents types de dispositifs, que ce soit un ordinateur de bureau, une tablette ou un smartphone. Elle est également réactive, s'adaptant aux différentes tailles d'écran pour offrir une expérience utilisateur optimale sur tous les supports.

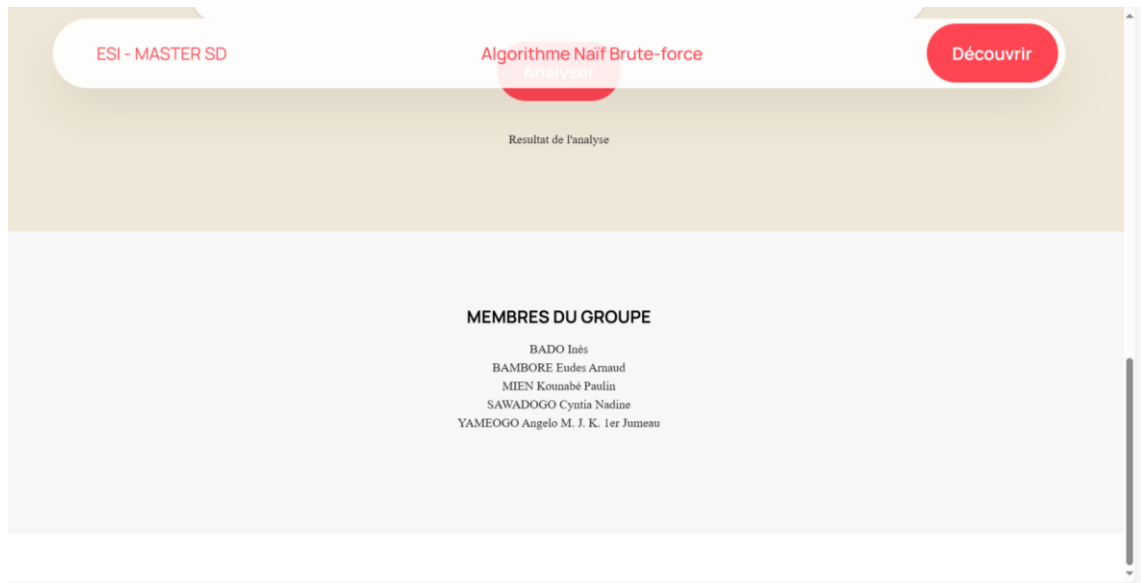
7. Présentation de l'application

❖ Page d'accueil

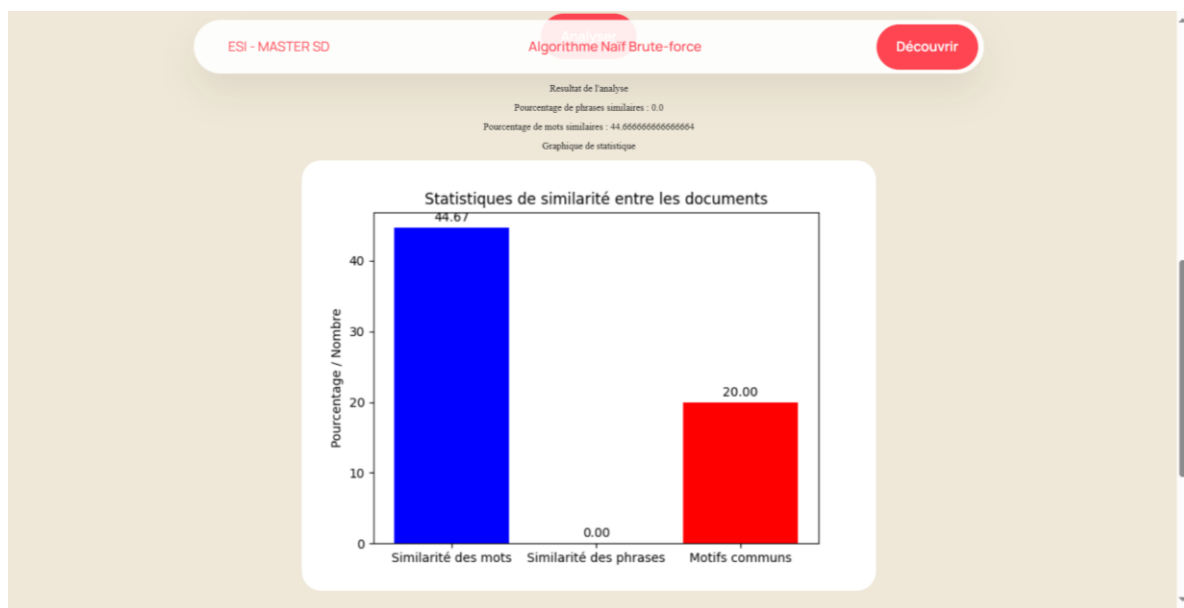


❖ Formulaire de saisie

The image shows the input form of the application. It has a light beige background. At the top, there is a horizontal header bar with 'ESI - MASTER SD' in red on the left, 'Algorithme Naïf Brute-force' in red in the center, and a red button with 'Découvrir' in white on the right. Below the header, the main heading 'Choisissez les deux fichiers' is written in large red font. Underneath, there are two identical input fields. Each field has a light blue button labeled 'Choisir un fichier' and a text area that says 'Aucun fichier n'a été sélectionné'. Below these, there is a single input field with the label 'Longueur du motif'. At the bottom center, there is a red button labeled 'Analyser'. Below the 'Analyser' button, the text 'Resultat de l'analyse' is visible in a small font.



❖ Résultats après l'analyse de deux documents



8. Résumé des objectifs atteints.

Nous avons réussi à :

- ❖ Implémenter l'algorithme avec Django.
- ❖ Utiliser un front-end pour l'affichage des données.
- ❖ L'application arrive à faire les analyses énumérées dès le départ, à savoir :
- ❖ La similarité des mots
- ❖ La similarité des phrases
- ❖ Les motifs communs

9. Conclusion

Ce projet nous a permis de plonger profondément dans le domaine du traitement automatique du langage naturel, en particulier dans l'analyse de similarité entre documents. Grâce à l'implémentation de l'algorithme naïf de Brute-force, nous avons pu développer une application capable de comparer efficacement des textes. Malgré les défis rencontrés, tels que la gestion de l'upload des documents et les divergences dans l'optimisation de l'algorithme, nous avons acquis une compréhension solide des concepts et techniques nécessaires. Ce projet a non seulement enrichi nos connaissances techniques, mais a également renforcé notre capacité à travailler en équipe et à résoudre des problèmes complexes de manière collaborative.