AMS PROJET 3

Ylies Chementel [1], *Mohamed Rouabhia*[2]

[1]Avignon Université, CERI, Avignon, France

ylies.chementel@alumni.univ-avignon.fr, mohamed.rouabhia2@alumni.univ-avignon.fr

Abstract

Dans cet article, on va se plonger dans l'extraction automatique de réseaux de personnages à partir d'un texte, en l'occurrence Fondation d'Isaac Asimov. L'idée derrière tout ça, c'est de construire un graphe qui représente les interactions entre les différents personnages du livre, un peu comme une carte des relations entre eux. Pour y arriver, on va passer par plusieurs étapes : d'abord, il faut identifier les personnages en reconnaissant leurs noms (c'est ce qu'on appelle la reconnaissance d'entités nommées), puis il faut résoudre les alias, c'est-à-dire associer les différents noms d'un même personnage. Ensuite, on cherche à détecter les interactions, par exemple en repérant quand deux personnages apparaissent dans des phrases proches. Enfin, on construit un graphe où chaque personnage est un noeud et chaque lien entre eux représente une interaction. On vous expliquera ici comment on a mis en place ce pipeline, les défis qu'on a rencontrés, et la qualité des réseaux qu'on a obtenus. L'objectif est non seulement de sortir un graphe fonctionnel, mais aussi d'en tirer des enseignements sur les relations entre les personnages et sur les possibles améliorations à apporter.

1. Méthodologie

1.1. Préparation des données

Dans cette section, nous expliquons comment la préparation des données a été effectuée pour le projet d'extraction de réseaux de personnages. L'objectif principal de cette préparation était de nettoyer et de normaliser les textes, afin de les rendre plus adaptés à l'extraction d'entités nommées.

1.1.1. Nettoyage du texte

Avant de passer à l'extraction des personnages, un nettoyage du texte était nécessaire pour le préparer à l'analyse. Ce nettoyage comprenait plusieurs étapes :

- **Suppression des retours à la ligne indésirables** : Les retours à la ligne (\n) ont été remplacés par des espaces, sauf dans les cas où le caractère précédant le retour à la ligne était un tiret (-), indiquant probablement une coupure de mot ou une continuation.
- **Remplacement des tirets longs** : Les tirets longs (—) ont été supprimés, car ils peuvent gêner dans l'extraction des entités nommées.
- **Formatage des ponctuations** : Des expressions régulières ont été utilisées pour ajouter des sauts de ligne après certains signes de ponctuation (point, point d'interrogation, point d'exclamation, parenthèses et guillemets) pour structurer les phrases de manière plus uniforme et faciliter l'extraction.

Ces modifications ont permis de rendre le texte plus cohérent et mieux structuré pour les étapes suivantes de l'extraction.

1.1.2. Traitement des personnages en majuscules

Une autre étape importante de ce pré-traitement consistait à normaliser les noms des personnages. Dans le texte, certains personnages étaient écrits en majuscules, mais parfois de manière incorrecte ou incohérente. Le processus suivant a été appliqué pour rectifier cela :

- **Détection des mots en majuscules** : Le texte a été parcouru pour identifier les mots en majuscules (qui sont souvent utilisés pour désigner les noms de personnages).
- **Vérification de la catégorie grammaticale** : Pour éviter les erreurs de transformation, chaque mot en majuscules a été analysé à l'aide du modèle NLP de Stanza. Seuls les mots en majuscules qui étaient classés comme des noms propres (PROPN dans les tags POS) ont été considérés comme des personnages potentiels.
- **Correction des majuscules** : Si un mot en majuscules correspondait à un nom propre, il était capitalisé correctement. Par exemple, "CLÉON" était transformé en "Cléon" pour correspondre aux algorithmes d'extraction utilisés.

Ce traitement permet de s'assurer que les noms des personnages sont traités de manière homogène dans tout le texte.

1.1.3. Enregistrement des textes traités

Une fois le texte nettoyé et les corrections effectuées, il a été sauvegardé dans un répertoire dédié à ces fichiers prétraités. Chaque chapitre traité est sauvegardé avec un nouveau nom de fichier dans le répertoire `Textes_Processed`, où les chapitres du livre sont réorganisés et prêts pour l'extraction des entités nommées.

1.2. Extraction des personnages

Cette section décrit le processus d'extraction des personnages du texte. Nous avons utilisé `Flair` pour différencier les lieux des personnages, appliqué des filtres POS pour éliminer les entités non pertinentes, puis extrait les entités de type "PER" avec `Stanza`. Enfin, un tri des entités a permis d'éliminer les erreurs et les noms composés parasites pour obtenir une liste précise de personnages.

1.2.1. Utilisation de Flair pour extraire les lieux

Avant de procéder à l'extraction des personnages, il est essentiel de différencier les entités de type "personne" des entités de type "lieu". Pour cela, nous avons utilisé `Flair`, un outil performant pour la reconnaissance d'entités nommées (NER), particulièrement adapté pour la détection des lieux en français. En utilisant `Flair`, nous avons pu identifier de manière fiable toutes les entités de type "LOC" (lieu), ce qui nous a permis de les retirer dès le départ de la liste des personnages.

Cela nous a évité les erreurs où des noms de lieux seraient faussement interprétés comme des noms de personnages. Cette approche garantit que la liste des personnages ne contiendra que des entités de type "personne", ce qui est essentiel pour la suite de l'analyse.

1.2.2. Filtre des entités via le POS

Une fois que les entités nommées ont été extraites, un filtre supplémentaire a été appliqué pour éliminer celles qui, bien que détectées comme des personnes, ne le sont pas en réalité. En effet, certains noms propres ou termes peuvent être mal interprétés comme des personnages, alors qu'il s'agit en réalité de termes communs, de concepts ou de lieux.

Le filtre repose sur l'analyse des catégories grammaticales (POS) des mots détectés. En étudiant les étiquettes grammaticales, nous avons pu éliminer celles qui appartiennent à des catégories non pertinentes comme les adpositions (`ADP`), les déterminants (`DET`), les verbes (`VERB`), ou les adjectifs (`ADJ`). De plus, les mots marqués comme des noms communs ou des noms pluriels ont également été éliminés, car ils ne correspondent généralement pas à des personnages.

Les mots qui sont étiquetés comme des noms propres (`PROPN`) ont été conservés, mais uniquement s'ils respectent certaines conditions, comme ne pas être associés à une autre catégorie grammaticale qui les rendrait non pertinents pour l'extraction de personnages. Ce filtre permet ainsi de réduire les faux positifs et d'affiner la liste des entités à conserver.

1.2.3. Extraction des entités nommées de type "PER"

L'étape suivante consiste à extraire spécifiquement les entités nommées de type "PER" (personnes). Pour cela, nous avons utilisé `Stanza`, une bibliothèque NLP reconnue pour ses capacités d'extraction d'entités nommées. `Stanza` a permis d'identifier toutes les entités humaines dans le texte, en les marquant comme des entités de type "PER".

Au cours de cette étape, les entités extraites ont été nettoyées, en particulier pour supprimer les caractères indésirables tels que les retours à la ligne. De plus, seules les entités d'une longueur suffisante ont été conservées, afin d'éviter l'inclusion de noms trop courts ou d'erreurs de reconnaissance (par exemple, des fragments de texte qui ne constituent pas un personnage).

1.2.4. Tri des entités lieux

Une fois les entités de type "PER" extraites, il est essentiel de s'assurer qu'aucun lieu n'a été inclus par erreur dans la liste des personnages. Certaines entités, bien que détectées comme des personnes par `Stanza`, peuvent être en réalité des noms de lieux. Par exemple, un nom de ville ou un lieu géographique peut être mal interprété comme une personne.

Pour résoudre ce problème, nous avons utilisé `Flair`, qui avait déjà extrait les lieux du texte, pour effectuer un tri supplémentaire. Toute entité de type "PER" qui était également présente dans la liste des lieux extraits par `Flair` a été supprimée de la liste finale des personnages. Cela garantit que seuls les vrais personnages restent dans la liste.

1.2.5. Tri des mots composés parasites

Une autre difficulté réside dans les noms composés de plusieurs mots, comme ceux qui peuvent être utilisés pour désigner un personnage. Par exemple, une entité composé de deux mot comme "Ma Moustache" peut échapper aux filtre précédents. Pour résoudre ce problème, un tri supplémentaire a été appliqué.

Les entités composées de plusieurs mots ont été vérifiées pour s'assurer qu'elles correspondent bien à des personnages. Si un nom composé de plusieurs mots était mal interprété ou ne correspondait pas à un personnage réel, il était retiré de la liste. Ce tri a permis de filtrer des entités "parasites", c'est-à-dire des combinaisons de mots qui, bien qu'ayant une structure similaire à des noms propres, ne faisaient pas référence à des personnages dans le contexte du texte.

1.2.6. Conclusion

En résumé, l'extraction des personnages a été réalisée en plusieurs étapes, allant de l'utilisation de `Flair` pour extraire les lieux, à l'application de filtres POS pour éliminer les entités non pertinentes, jusqu'à l'extraction des entités de type "PER" et le tri final des mots composés parasites. Ces étapes permettent d'obtenir une liste précise et cohérente de personnages, en éliminant les erreurs courantes comme la confusion entre personnages et lieux ou entre noms propres et termes communs.

1.3. Résolution des alias

Dans cette section, nous avons détaillé les différentes étapes pour résoudre les alias des personnages. Cela inclut le rassemblement des personnages ayant le même nom de famille, l'ajout des prénoms isolés, et la séparation des personnages de la même famille mais avec des prénoms différents. Enfin, nous avons extrait tous les noms des chapitres du corpus pour mieux gérer les alias et éviter les doublons.

1.3.1. Rassemblement des personnages avec les mêmes noms de famille

Lors de l'extraction des personnages, plusieurs entités peuvent partager des noms similaires, notamment des personnages ayant le même nom de famille. Dans cette étape, nous avons regroupé ces personnages en fonction de leurs noms de famille. Cela permet de rassembler toutes les variantes d'un même personnage, facilitant ainsi la gestion des alias.

1.3.2. Rajout des prénoms isolés dans une liste

Il arrive que certains personnages, n'aient pas été correctement rassemblés. Ces prénoms, souvent isolés, ont été ajoutés dans une liste séparée pour éviter qu'ils ne soient oubliés ou mal catégorisés. Cela permet de compléter les informations sur les personnages en associant les prénoms aux noms de famille pertinents.

1.3.3. Séparation des personnages de la même famille mais pas le même prénom

Après avoir rassemblé les personnages par nom de famille, il était important de les différencier si leurs prénoms étaient différents. Par exemple, deux personnages de la famille "Baley", tels que "Elijah Baley" et "Bentley Baley", doivent être considérés comme deux entités distinctes, malgré leur nom de famille commun. Cette étape permet donc de séparer les personnages ayant un nom de famille identique, mais un prénom différent, afin d'éviter toute confusion.

1.3.4. Suppression des listes qui sont des sous-listes d'autres

Au fur et à mesure des étapes de traitement, des doublons peuvent apparaître sous la forme de sous-listes. Par exemple, un

groupe de personnages peut être contenu dans un autre groupe plus large. Cette étape vise à éliminer ces sous-listes redondantes, garantissant ainsi que chaque ensemble de personnages reste unique et représentatif d'une entité distincte. En supprimant ces doublons, on simplifie la gestion des alias et on évite de multiplier les références aux mêmes personnages sous des formes différentes.

1.3.5. Extraction de tous les noms de tous les chapitres du corpus

Afin de mieux gérer les alias, il a été décidé d'extraire toutes les entités nommées présentes dans l'ensemble du corpus, plutôt que de les séparer par chapitre. Cette approche permet de relier des personnages mentionnés dans différents chapitres sous des noms différents. Par exemple, des personnages appelés "Dors" et "Venabili" peuvent apparaître séparément dans différents chapitres, mais il peut être évident qu'ils se réfèrent à la même personne lorsqu'on considère l'ensemble du livre. En regroupant toutes les entités nommées dans une seule liste, nous avons pu mieux détecter et traiter les alias.

Cette méthode permet également de résoudre des problèmes où des personnages sont mentionnés sous des variantes de leur nom dans différents chapitres, mais sans lien explicite dans ces chapitres. En utilisant cette liste complète, nous avons pu détecter et rassembler les alias de personnages qui, bien que séparés dans les chapitres, font référence à la même personne.

1.4. Relations entre les personnages

Dans cette section, nous présentons la méthode utilisée pour identifier les relations entre les personnages dans le texte. Ces relations sont déterminées en fonction de la proximité des personnages dans les différents passages, permettant ainsi de lier les personnages entre eux selon les interactions ou mentions communes.

1.4.1. Identification des positions des personnages

Afin d'établir des relations entre les personnages, il est nécessaire de localiser leurs mentions dans le texte. Pour ce faire, nous analysons les tokens du texte et recherchons les mentions de chaque personnage, en tenant compte de toutes ses variantes (alias). Chaque mention d'un personnage est associée à sa position dans le texte. Ces positions sont stockées dans un dictionnaire, où chaque clé correspond à un personnage et chaque valeur contient une liste des indices des tokens où ce personnage est mentionné.

1.4.2. Création des relations entre les personnages

Pour établir les relations entre les personnages, un dictionnaire est utilisé pour stocker les relations uniques qui existent dans le texte. Ce dictionnaire associe chaque personnage à un autre personnage avec lequel il entretient une relation, en prenant en compte non seulement les noms complets des personnages, mais aussi leurs variantes. Par exemple, si l'entité "Baley" est en relation avec "Sarton", et que l'entité "Elijah" (qui peut aussi être appelée "Elijah Baley") est également en relation avec "Sarton", alors le dictionnaire indiquera la relation suivante : `'Sarton': ['Baley', 'Elijah Baley']`. Cela permet de regrouper les relations en tenant compte des alias ou variantes des noms, facilitant ainsi l'analyse des interactions entre les personnages.

Une fois les positions des personnages identifiées dans le texte, les relations peuvent être établies. Si deux personnages apparaissent à une distance relativement courte dans le texte (par exemple, moins de 25 tokens), une relation est considérée comme existante entre eux. Cette distance permet de capturer des interactions réelles entre les personnages, reflétant ainsi les liens narratifs. En procédant de la sorte, un lien direct est créé entre les personnages qui apparaissent proches l'un de l'autre, renforçant ainsi la pertinence de leur association dans le contexte narratif. Cette méthode permet de visualiser les relations entre les personnages en fonction de leur proximité et d'identifier des dynamiques clés dans l'histoire.

1.4.3. Dictionnaire des relations sous forme de liste

Un second dictionnaire est généré à partir du dictionnaire de relation unique pour organiser les relations sous forme de listes . Ce dictionnaire associe chaque personnage à une liste d'autres personnages avec lesquels il est lié. Cela permet de mieux structurer et de visualiser l'ensemble des relations entre les personnages, facilitant ainsi leur import dans le graphe à crée.

1.5. Construction du réseau

Une fois les relations entre les personnages établies et stockées dans le dictionnaire `dictionnaireRelationsListe`, l'objectif est de construire un graphe représentant ces relations, où chaque personnage est un nœud et les relations sont des arêtes. Chaque arête est associée à un poids, indiquant la force ou la fréquence de la relation.

1.5.1. Initialisation du graphe

Le graphe est initialisé en créant un nœud pour chaque personnage et en ajoutant des arêtes représentant les relations entre les personnages. Si une relation existe déjà entre deux personnages, le poids de l'arête est incrémenté, ce qui reflète la fréquence des interactions. Si l'arête n'existe pas encore, une nouvelle arête est ajoutée avec un poids initial de 1.

1.5.2. Gestion des alias des personnages

Pour chaque groupe d'alias d'un personnage, un nœud est ajouté au graphe. Le premier élément du groupe d'alias est considéré comme le nom principal, et les autres variantes sont jointes sous forme de variantes séparées par un point-virgule. Cette approche permet de regrouper différentes formes d'un même nom tout en conservant les informations sur les alias, garantissant ainsi que toutes les références à un personnage sont correctement associées.

1.5.3. Attribution des attributs aux nœuds

Une fois que tous les nœuds et arêtes ont été ajoutés, des attributs sont associés à chaque nœud. Par exemple, si un nœud n'a pas encore l'attribut `names`, cet attribut est créé et assigné avec le nom principal du personnage, accompagné éventuellement de ses variantes.

1.5.4. Enrichissement des informations sur le graphe

Après la construction du graphe, des informations supplémentaires, telles que les identifiants des chapitres et les représentations du graphe au format GraphML, sont ajoutées à une structure de données. Ces informations permettent d'analyser et de visualiser les relations entre les personnages dans le réseau, tout en fournissant une compréhension

plus approfondie des dynamiques qui se dessinent au sein de l'histoire.

Ainsi, cette méthode permet de créer un réseau dynamique de personnages et de leurs relations, facilitant l'exploration des interactions et des liens narratifs au sein du texte.

2. Résultats

2.1. Analyse du réseau des relations entre les personnages

Nous sommes assez satisfaits de nos résultats car nous avons réussi à minimiser les entités nommées qui ne sont pas des personnages via les divers filtres, et avons relativement bien optimisé notre heuristique pour rassembler les alias. Notre meilleure note sur le leaderboard Kaggle était de 0.51 (équipe "Vulcains") avec 70 soumissions, ce qui est le maximum de soumissions pour l'ensemble des équipes. Nous avons été une équipe qui a avant tout persévéré malgré les différents problèmes rencontrés.

Nous avions encore de nombreuses idées à implémenter, telles que la Coreference ou la possibilité de visualiser notre graphe, mais nous avons malheureusement manqué de temps.

Cela dit, certains problèmes persistent, comme des entités "parasites" qui ne devraient pas être présentes, telles que "Poliment" ou "Tabou", et qui ne peuvent pas encore être éliminées. Malgré cela, nous sommes satisfaits des résultats obtenus pour ce premier semestre.

2.2. Problèmes et Erreurs

Au cours de ce projet, plusieurs défis et problèmes ont été rencontrés, qui ont entravé le bon déroulement des tâches et affecté la qualité des résultats obtenus.

- **Problèmes de mémoire :** Les librairies utilisées dans ce projet ont nécessité une grande quantité de ressources, notamment en termes de mémoire. Par exemple, la librairie `Stanza` consommait une telle quantité de mémoire que cela provoquait l'arrêt de l'ordinateur. Dans d'autres cas, le programme pouvait tourner pendant plus de 30 minutes avant de donner un résultat, ce qui n'était pas optimal pour la détection d'erreurs.
- **Problèmes d'extraction de noms :** Nous avons cherché à optimiser le processus d'extraction des noms, mais certaines entités ont échappé à la détection. Un exemple de ce type de problème est l'entité "Dieu", qui apparaît dans des phrases comme "Dieu merci, il est vivant". Bien que "Dieu" ne soit pas une entité dans ce contexte, nous n'avons pas trouvé de solution satisfaisante pour gérer ce genre de cas.
- **Limite des modèles de NLP :** Les modèles de traitement du langage naturel (NLP) comme `Stanza`, `Flair`, `Spacy`, etc., présentent plusieurs limites. Bien que `Stanza` ait été le modèle le plus performant pour l'extraction des personnages, il reste néanmoins faillible, laissant passer des entités problématiques. `Flair` est utile, mais il présente aussi des défauts, notamment en détectant mieux les lieux que les personnages. Cela est dû à une limitation des modèles NLP, particulièrement en français, qui est une langue plus complexe à analyser que l'anglais. De plus, le nombre limité de modèles pré-entraînés en français pour la tâche de *token classification* sur des plateformes comme Hugging Face rend le choix des bons modèles difficile. En effet, parmi les 18 modèles d'entraînement français disponibles, seulement 5 s'alignent véritablement avec les besoins de ce projet, sans oublier que la plupart n'ont pas de documentation associée et que leur taux de précision reste faible. Toutes ces difficultés soulignent qu'un modèle NLP est relativement facile à trouver, mais qu'il est bien plus complexe à implémenter correctement et à adapter aux besoins spécifiques d'un projet comme celui-ci.
- **Les alias, un problème complexe :** Nous avons essayé plusieurs approches pour gérer les alias, mais aucune n'a montré une efficacité aussi bonne que l'approche heuristique. Les méthodes d'embedding sont plus légères mais ne sont pas aussi précises, et parfois elles regroupent des entités qui, bien que similaires en termes de lettres, ne sont pas liées. Par exemple, des méthodes comme `DBSCAN` (clustering basé sur la densité) ont montré de bons résultats, mais certaines entités, telles que "Wellis" et "Lieutenant Alban Wellis", ne sont pas regroupées correctement en raison de la longueur du second nom. De même, l'utilisation de différentes mesures de similarité entre chaînes de caractères (Jaro-Winkler, Levenshtein, Damerau-Levenshtein) n'a pas permis d'obtenir des résultats satisfaisants. Par exemple, bien que "Elijah Baley" et "Lije Baley" désignent la même personne, ces méthodes ont souvent associé "Lizzie" et "Lije", ce qui est incorrect.
- **Problème des entités totalement différentes mais désignant la même personne :** Un autre problème majeur lié aux alias est la gestion des noms complètement différents qui désignent la même personne, comme "Cléon" et "Empereur", ou encore "Chetter Hummin" et "Eto Demerzel". Bien que nous ayons consulté plusieurs articles et programmes GitHub, aucune méthode efficace n'a été trouvée pour résoudre ce problème.
- **Nettoyage du texte :** Le nettoyage du texte a représenté un défi majeur. Parfois, laisser le texte brut sans nettoyage a été plus bénéfique que de le nettoyer, car cela dépendait fortement de l'interaction avec les modèles NLP utilisés. De plus, l'utilisation de plusieurs modèles NLP rendait le nettoyage plus complexe. Dans certains cas, des coupures dans le texte ou des phrases incomplètes ont rendu l'extraction encore plus difficile, car cela privait le modèle du contexte nécessaire.
- **Limite de token pour la co-occurrence :** La limite de 25 tokens pour définir une co-occurrence semble insuffisante. Une telle limite ne représente pas une fenêtre contextuelle adéquate pour établir des relations entre personnages. Avant de prendre conscience de cette limite, nous avons utilisé `Gensim` pour analyser l'ensemble du chapitre et déterminer les relations via une probabilité allant de 0 à 1. Nous avons sélectionné uniquement les relations avec une probabilité de 0.9, garantissant des relations pertinentes et présentes dans le texte. Nous pensons qu'en étendant l'analyse à l'ensemble du chapitre, sans la contrainte des 25 tokens, on pourrait obtenir un graphe plus exhaustif.
- **Problèmes dans la détection des personnages en français :** Un problème supplémentaire réside dans la détection des personnages dans des contextes où des noms d'origine anglaise sont utilisés. Les modèles entraînés sur des textes en français ont du mal à détecter des lieux comme "SpaceTown", qui paraissent évidents

en raison du suffixe "Town". Cependant, ces modèles peinent à identifier correctement ce type d'entité, ce qui constitue une limitation importante.

3. Améliorations et Perspectives

- **Créer et entraîner notre propre modèle :** Comme mentionné précédemment, l'utilisation de modèles pré-entraînés sur des données qui ne correspondent pas à notre corpus pose un problème majeur pour l'extraction correcte des entités. Ainsi, si l'opportunité se présente, nous pensons que l'entraînement de notre propre modèle sur ce corpus spécifique permettrait d'optimiser considérablement l'extraction des personnages tout en réduisant l'impact des lieux et autres entités parasites. Ce modèle unique permettrait également de gagner en efficacité, à la fois en termes de temps de compilation et d'implémentation.
- **Utiliser la *Coreference* :** Au cours de nos recherches, nous avons découvert que la *coreference* (référence croisée) pourrait être une méthode prometteuse pour lier des termes comme "Sire" et "Cléon". Nous avons identifié que des outils tels que `Stanza`[1] et `Spacy`[2] disposaient de modules dédiés à cette technique. Après avoir réussi à les faire fonctionner, nous espérons les utiliser pour améliorer la détection des alias et rendre notre modèle plus performant dans ce domaine.
- **Améliorer notre heuristique pour la recherche d'alias :** Bien que notre approche heuristique pour la détection des alias soit fonctionnelle, elle reste assez rudimentaire et peu optimisée. Nous avons l'intention de travailler sur l'amélioration de ces algorithmes au prochain semestre pour augmenter la précision et l'efficacité de cette méthode.
- **Améliorer le système de poids sur les relations :** Nous avons débuté l'implémentation d'un système de poids pour évaluer la force des relations entre les personnages, mais ce système est loin d'être optimal. Nous prévoyons de l'affiner et de l'améliorer lors du prochain semestre, afin d'ajouter plus de pertinence et de précision à l'analyse des relations.
- **Prédiction des liens et du degré d'amitié ou inimitié entre les personnages :** Concernant la prédiction des liens affectifs entre les personnages, nous avons plusieurs pistes intéressantes. Par exemple, nous pensons que les personnages appartenant à la même famille pourraient être facilement catégorisés comme "famille" lorsqu'ils sont en relation dans le texte. Nous envisageons même d'implémenter automatiquement cette relation familiale lorsqu'ils apparaissent dans le même chapitre.

  Pour les liens d'amitié, nous avons exploré plusieurs approches et pris connaissance de modèles [3] capables d'analyser le sentiment d'un message. Nous pensons qu'un tel modèle pourrait être utilisé pour évaluer le degré d'amitié entre les personnages, en fonction du ton émotionnel d'une phrase. Par exemple, une phrase comme "Elijah déteste cette personne" permettrait de déterminer une relation négative en analysant l'émotion véhiculée.

  De plus, nous pensons que cette analyse de sentiment pourrait être combinée avec la *coreference* pour améliorer la précision de l'analyse. En effet, si l'on se limite uniquement aux noms pour déterminer une relation, cela pourrait être insuffisant. Imaginons par exemple la phrase "Elijah déteste cette personne". La *coreference* nous permettrait d'identifier à qui se réfère "cette personne", et l'analyse de sentiment nous aiderait à évaluer si cette relation est amicale ou hostile. Avec l'intégration de ces différentes méthodes, il nous semble que nous pourrons prédire de manière fiable le degré d'amitié entre les personnages.
- **Visualiser le graphe :** Enfin, un objectif que nous n'avons pas pu atteindre ce semestre, mais qui reste sur notre liste de priorités, est la visualisation du graphe des relations entre les personnages. Nous avons quelques idées sur la manière de procéder et espérons pouvoir réaliser cette visualisation au prochain semestre. Cela permettrait non seulement de rendre notre analyse plus tangible, mais aussi d'offrir une meilleure compréhension visuelle des liens complexes entre les entités du texte.

[1] `https://stanfordnlp.github.io/stanza/coref.html`

[2] `https://spacy.io/universe/project/coreferee`

[3] `https://huggingface.co/nlptown/bert-base-multilingual-uncased-sentiment?utm_source=chatgpt.comhttps://www.kaggle.com/code/antoinetheissen75/projet-analyse-de-donn-e-r-seaux-sociaux`

4. References

Voici tous les liens qui nous ont été utiles ou que nous avons visités pour obtenir plus d'informations lors de ce projet :

- `https://stanfordnlp.github.io/stanza/`
- `https://spacy.io/`
- `https://flairnlp.github.io/`
- `https://radimrehurek.com/gensim/`
- `https://www.nltk.org/`
- `https://huggingface.co/GEODE/bert-base-french-cased-edda-ner-joint-label-lev`
- `https://huggingface.co/Pclanglais/French-TV-transcript-NER`
- `https://huggingface.co/flair/ner-french`
- `https://huggingface.co/dbmdz/bert-base-french-europeana-cased`
- `https://huggingface.co/Jean-Baptiste/camembert-ner`
- `https://www.kaggle.com/code/antoinetheissen75/projet-analyse-de-donn-e-r-seaux-sociaux`
- `https://journals.openedition.org/resf/1183`
- `https://github.com/thunlp/OpenNRE`
- `https://www.youtube.com/watch?v=ioGry-89gqE`
- `https://machinelearningmastery.com/what-are-word-embeddings/`
- `https://p.migdal.pl/blog/2017/01/king-man-woman-queen-why`
- `https://aclanthology.org/D15-1088.pdf`