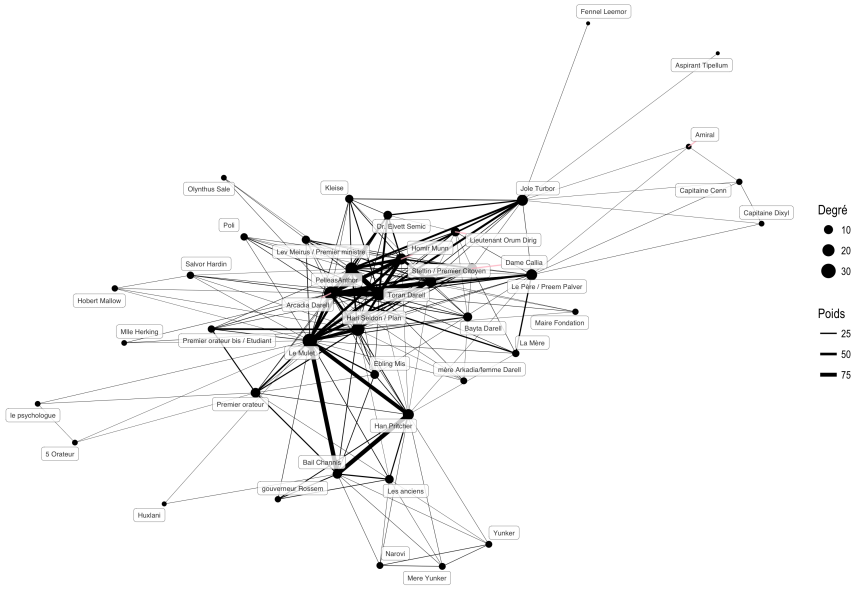
Extraction automatique de réseaux de personnages : Fondation d'Asimov (V2)

Juan-Manuel Torres et Arthur Amalvy

15 novembre 2023

Table des matières



1 Extraction automatique de réseaux de personnages

Extraire un réseau de personnage automatiquement dans un texte nécéssite de résoudre un certain nombre de tâches de traitement des langues naturelles (TALN) [1] :

1. Identification des personnages :
   (a) Détection des occurrences des personnages, qui peut se traduire par la *reconnaissance d'entités nommées* (REN).
   (b) Unification de ces occurrences (pour chaque occurrence, à quel personnage appartient-elle ?), qui dans notre cas peut se traduire par la tâche de *résolution d'alias*.
2. Détection des interactions entre personnages. Dans ce défi, nous nous limiterons aux *co-occurrences entre personnages* (mais vous êtes libres d'explorer d'autres formes d'interactions dans votre rapport si vous le souhaitez).
3. Construction du graphe.

Généralement, l'extraction de réseaux de personnages s'effectue à l'aide d'un pipeline implémentant ces différentes tâches séquentiellement.

N'hésitez pas à vous référer à [1] pour plus de détails sur ces différentes étapes : l'article présente une revue exhaustive de l'extraction de réseaux de personnages dans les oeuvres de fiction. Les sous-parties suivantes décrivent la réalisation des différentes étapes mentionnées.

1.1 Identification des personnages

1.1.1 Reconnaissance d'entités nommées

La tâche de REN est une tâche classique et bien étudié en TALN (voir NLPProgress). Il s'agit de détecter l'apparition explicite d'entités nommées dans un texte, ainsi que le type de ces entités. Par exemple, dans le texte suivant :

```
Hari Seldon habitait à Trantor.
```

On peut distinguer deux entités : "Hari Seldon", de type `personne`, et "Trantor", de type `lieu`. Dans l'exercice d'extraction de réseaux de personnages, seules les entités de type `personne` nous intéréssent. Pour représenter le problème, un schéma classique est le BIO :

```
Hari  Seldon habitait à Trantor
B-PER I-PER  O        O B-LOC
```

Les début d'entités sont marqués par B-CLASSE, et les tokens à l'intérieur des entités sont marqués I-CLASSE. Les autres tokens sont notés O (Others). Ce schéma permet de représenter le problème comme de classification de token, mais d'autres représentations sont possibles. À vous d'expérimenter !

De nombreuses librairies existent pour la REN, n'hésitez donc pas à vous en servir.

1.1.2 Résolution d'alias

La REN décrite ci-dessus permet d'extraire les apparitions des personnages, mais pas nécessairement de savoir à quel personnage elles font référence ! Comment savoir que les expressions "Mr. Seldon" ou "Seldon" désignent le même personnage `HARI SELDON` ?

Ce problème peut être partiellement résolut grâce à la tâche de *résolution d'alias*. Il s'agit, en prenant comme entrée une liste de noms de personnages, de grouper ces noms afin que chaque groupe représente tous les noms d'un personnage. Par exemple, avec les noms suivant :

```
{John, John Traitor, Hari, Hari Seldon, Mr Seldon,
 Sir Traitor, Mme Seldon}
```

On souhaite obtenir les groupes suivants, qui correspondent chacun à personnage :

```
{John, John Traitor, Sir Traitor},
{Hari, Hari Seldon, Mr Seldon},
{Mme Seldon}
```

Si le problème semble simple, il est en réalité plus complexe qu'il n'y parait. Vous pouvez vous référer à [2] pour un exemple de système à base de règles qui permet de résoudre ce problème.

1.2 Détection d'interactions : Co-occurrences

Dans ce défi, nous allons considérer qu'une interaction entre deux personnages arrive lorsque deux apparitions de ceux-ci sont "proches" dans le texte. Par soucis de simplicité, dans ce sujet, nous allons considérer que deux entités sont en co-occurence lorsqu'elles apparaissent à une distance arbitraire de **25 tokens ou moins**.

1.3 Extraction du graphe

Une fois les interactions entre personnages détectées, il est facile d'extraire un graphe avec celles-ci. Chaque noeud représentera un personnage obtenu à l'étape de résolution d'alias, et chaque lien représentera le fait que deux personnages ont eut au moins une interaction. Le poids des liens représentera le nombre de co-occurences entre les deux personnages.

Si vous utilisez Python, nous vous conseillons la librairie networkx afin de manipuler et d'exporter vos graphes.

2 Sujet du défi

Le défi consiste à extraire des réseaux de personnages sur le corpus Fondation fourni. Pour cela, vous devrez concevoir et programmer un pipeline séquentiel permettant d'extraire des réseaux de personnages.

Il faudra à la fois :

— Extraire un réseau de personnages par livre du corpus Fondation, et tenter de commenter qualitativement la qualité des réseaux extraits.

— Extraire un réseau de personnages par chapitre du corpus Fondation d'évaluation afin de participer au leaderboard (voir Section 2.1.1).

Conseil important : vous êtes encouragés à commencer par construire une baseline le plus vite possible afin d'avoir des résultats rapidement pour les améliorer ensuite progressivement, quitte à implémenter certaines étapes de manière triviale initialement.

2.1 Notation

La notation se fera en fonction du tableau suivant :

Partie	Pourcentage de la note
QCM	20%
Article	20%
Code et réalisation générale	20%
Leaderboard	20%
Présentation orale	20%

2.1.1 Leaderboard

Une partie de la notation proviendra du classement des groupes sur un leaderboard Kaggle. Afin d'éviter tout problème de triche, l'évaluation se fera sur des chapitres aléatoires du corpus d'évaluation [1].

Les graphes devront être extraits au format `graphml`, la soumission au leaderboard se fait ensuite sous format csv (Voir Section 3.2). Chaque noeud devra avoir un attribut `names`, qui contiendra les différents noms des personnages (comme indiqué dans la partie "résolution d'alias") séparés par des virgules.

Les métriques d'évaluation seront les suivantes :

F1-score des noeuds Le F1-score des noeuds sera calculé en se basant sur [2], en prenant en compte l'attribut "names" de chaque noeud.

F1-score des arêtes On calculera le F1-score des arêtes en supposant un alignement entre les noeuds de votre graphe prédit et les noeuds du graphe de référence, obtenu en calculant la similarité entre les noeuds. La précision et le rappel seront calculés avec les définitions suivantes :

1. Notre capacité d'annotation étant limitée, cette configuration permet d'éviter vous annotiez à la main le peu de chapitres que nous avons annotés pour obtenir des scores parfaits.

Vrai positif Une arête entre deux noeuds est prédite et apparaît dans le graphe de référence.

Faux positif Une arête entre deux noeuds est prédite mais n'apparaît pas dans le graphe de référence.

Vrai négatif Une arête entre deux noeuds n'est pas prédite et n'apparaît pas dans le graphe de référence.

Faux négatif Une arête entre deux noeuds n'est prédite mais apparaît dans le graphe de référence.

Voir la Section 3.3 pour plus de détails sur le calcul des métriques.

2.1.2 Article et code

Un article de 4 pages devra être fourni par chaque équipe, reprenant les principaux points techniques et résultats de vos travaux. Il devra également expliciter les erreurs produites par vos algorithmes et leurs raisons. Cet article doit être écrit en LaTeX et formatté grâce au style Interspeech fourni. Le code devra être rendu public (sur une plateforme comme GitHub out GitLab) et avoir un README expliquant au minimum comment lancer le programme permettant d'extraire un réseau pour soumission au leaderboard. Plus de détails sont disponibles dans le cours d'introduction de l'ECUE. N'oubliez pas qu'il faudra aussi évaluer les articles de 2 autres équipes (bonus/malus de +2/-2 sur la note finale) !

Des points bonus seront attribués en cas d'exploration de thèmes avancés non explicitement demandés par le sujet. Voici quelques exemples de thèmes possibles :

— Amélioration de performance sur les tâches de NLP du pipeline
— Prise en compte de la polarité des relations par l'extraction d'un réseau signé
— Extraction de réseaux dynamiques

De manière générale, votre créativité sera récompensée par le barême !

2.1.3 Présentation orale

Comme pour les autres sujets, une présentation orale (15 minutes + 5 minutes de questions) sera de mise.

Références

[1] V. Labatut et X. Bost. "Extraction and Analysis of Fictional Character Networks : A Survey". In : *ACM Computing Surveys* 52 (2019), p. 89. doi : 10.1145/3344548.

[2] H. Vala et al. "Mr. Bennet, his coachman, and the Archbishop walk into a bar but only one of them gets recognized : On The Difficulty of Detecting Characters in Literary Texts". In : *Conference on Empirical Methods in Natural Language Processing*. 2015, p. 769-774. doi : 10.18653/v1/D15-1088.

3 Annexes

3.1 Détails sur l'annotation du jeu de données d'évaluation

3.1.1 Entités nommées

les entités nommées de type "personne" sont annotées en suivant les guidelines suivantes :

- Les titres précédant les noms font parties des entités (`Mr Bennet` est annoté `B-PER I-PER`)
- Les noms se référants à des familles sont annotés comme des personnes (`les Balkanys` est annoté `B-PER I-PER`)
- Les ethnonymes et démonymes ne sont pas annotés comme des personnes (`les Spaciens` est annoté `O O`)
- Les surnoms rendus évidents par la typographie sont annotés comme des personnes (`Tu-Sais-Qui` est annoté `B-PER`)
- Les entités discontinues sont annotées de manière discontinues (`Mr et Mme Seldon` est annoté `B-PER O B-PER I-PER`)
- Les noms communs capitalisés (comme `le Maire`, qui est annoté `B-PER I-PER`) sont annotés comme des personnes lorsqu'ils se réferent à des personnes précises, et non pas à des fonctions.
- Les surnoms suivants les personnages sont annotés comme des personnes (`Gandalf le Gris` est annoté `B-PER I-PER I-PER`)

3.2 Soumission au leaderboard

La soumission au leaderboard Kaggle se fait sous forme d'un fichier csv. Chaque ligne du CSV représente le graphe d'un chapitre du corpus leaderboard. Les colonnes de celui-ci doivent être les suivantes :

ID un ID unique identifiant le chapitre correspondant au graphe fourni. L'id est de la forme `{code du livre}{numéro du chapitre}`. Les numéros des chapitres démarrent à 0. Les codes de livres sont `paf` et `lca` (pour *Prélude à Fondation* et *Les Cavernes d'Acier*, respectivement). Ainsi, le graphe du premier chapitre de Prélude à Fondation a pour `ID` `paf0`, et celui du dernier chapitre des *Cavernes d'Acier* a pour `ID` `lca17`.

graphml le graphe du chapitre identifié par la colonne `ID`, au format graphml. Chaque noeud peut être n'importe quelle chaîne de caractères, mais un attribut `names` doit être présent contenant les noms du personnages apparaissant durant le chapitre. Ces noms doivent être séparés par des points-virgules (exemple : `Hari;Hari Seldon`)

Pour vous aider, voici un exemple de programme Python permettant exportant une soumission de test au format CSV :

```python
import networkx as nx
import pandas as pd


# (chapitres, code du livre)
books = [
    (list(range(0, 19)), "paf"),
    (list(range(0, 18)), "lca"),
]

df_dict = {"ID": [], "graphml": []}

for chapters, book_code in books:

    for chapter in chapters:

        G = nx.Graph()

        # Crée implicitement deux noeuds ("Hari" et "Dors"),
        # et ajoute un lien entre eux.
        G.add_edge("Hari", "Dors")

        # On ajoute les attributs "names"
        G.nodes["Hari"]["names"] = "Hari Seldon;Hari"
```

```
        G.nodes["Dors"]["names"] = "Dors;docteur Dors"

        df_dict["ID"].append("{}{}".format(book_code, chapter))

        graphml = "".join(nx.generate_graphml(G))
        df_dict["graphml"].append(graphml)

df = pd.DataFrame(df_dict)
df.set_index("ID", inplace=True)
df.to_csv("./my_submission.csv")
```

3.3 Métrique du leaderboard : détails

Le score du leaderboard est défini comme la moyenne du F1-score des noeuds et du F1-score des arêtes.

3.3.1 F1-score des noeuds

Soit un ensemble de noeuds de référence G et un ensemble de prédictions E. Chaque noeud de ces ensembles est lui-même un ensemble d'alias pour un personnage. On calcule la précision comme dans [2], en trouvant le couplage biparti maximal entre les ensembles de noeuds. La similarité entre deux ensembles de noms est donnée par :

$$sim_p(E_i, G_j) = 1 - \frac{|E_i - G_j|}{|E_i|} \tag{1}$$

Ainsi, les noms prédits en trop sont pénalisés. En notant f la fonction qui à E_i associe le noeud correspondant dans G (ou l'ensemble vide si aucun noeud ne correspond), on calcule :

$$Precision = \frac{\sum_i sim_p(E_i, f(E_i))}{|E|} \tag{2}$$

Pour le rappel, on procède de manière similaire. On calcule la similarité entre deux ensembles de noms ainsi :

$$sim_r(E_i, G_j) = \begin{cases} 1, & \text{Si } E_i \cap G_j \neq \emptyset \\ 0, & \text{Sinon} \end{cases} \tag{3}$$

En notant g la fonction qui à G_j associe le noeud prédit E_i (ou l'ensemble vide si aucun noeud ne correspond), Le rappel est ensuite :

$$Rappel = \frac{\sum_j sim_r(g(G_j), G_j))}{|G|} \tag{4}$$

3.3.2 F1-score des arêtes

Soit un graphe de référence $H = \{G, R\}$ et un graphe prédit $I = \{E, P\}$. Afin de calculer le F1-score des arêtes, on procède d'abord à un couplage biparti maximal entre les noeuds de H et de I, similairement à ce que nous avons décrit pour le F1-score des noeuds. La similarité entre deux noeuds est donné par l'index de Jaccard entre leurs noms associés. Grâce à ce couplage, on peut également définir un couplage entre les ensembles d'arêtes R et P. On note h la fonction de couplage allant de R à $P \cup \{\varnothing\}$, et m la fonction allant de P à $R \cup \{\varnothing\}$. Ainsi, on peut calculer :

$$Precision = \sum_i \frac{|\{P_i : m(P_i) \in R\}|}{|P|} \tag{5}$$

$$Rappel = \sum_j \frac{|\{R_j : h(R_j) \in P\}|}{|R|} \tag{6}$$