[WIA] Rapport de l'application

Jonathan Boaknin

Décembre 2019

Table des matières

1 Objectif de l'application

Le projet porte sur le thème de « l'extraction de relations ». Le but global du programme est d'extraire les différents types de relations qui existent entre des personnages fictifs d'une narration ou d'une histoire. En d'autres termes, à partir des personnages que l'on a clairement définis, le programme va tenter d'extraire les relations qui existent entre eux à partir d'une même catégorie syntaxique.

Par exemple, à partir de la phrase « Alice Lee est la soeur de Henry », on peut extraire les personnages « Alice Lee » et « Henry » pour ensuite déterminer leur catégorie de relation par « Famille », car le mot qui se trouve entre les deux personnages (« soeur ») est un champs lexical de « famille ». On définit ainsi le triplet (Alice Lee,Famille, Henry) qui traduit la relation entre Alice Lee et Henry. Ce triplet correspond dans notre contexte à une relation de personnages. L'application mise en oeuvre va donc essayer d'extraire une série de relations comme celles-ci et les afficher en sortie.

2 Choix du corpus

Les textes narratifs utilisés pour tester le programme proviennent tous du site Projet Gutenberg qui recensent un très grand nombre de récits, tous libres de droits. On a donc pris 4 récits tels que « Sherlock », « Little womens », « Main Street »et « Middlermarch. Il y un cinquième texte du nom de « debug.txt », qui a permis de tester le programme avant de le soumettre à de vrais récits. Ces récits ont été amputés de quelques chapitre car l'exécution du programme devenait trop long, donc compliquer à débugger. Il y a eu aussi des problèmes d'encodage, essentiellemnt avec les tiret, les guillemets et les apostrophes qui ont été remplacés par un encodage correcte, à la main (en faisant rechercher -> remplacer).

De plus, les extraits du corpus du Projet Gutenberg n'étaient pas annotés, il a donc fallu le faire manuellement, en définissant le lien entre chaque personnage de cette façon :

Alice Lee,Family,Henry
Alice Lee,Social,Jean

Chaque champ lexical est défini dans un fichier .txt (relations_properties.txt), avec la catégorie syntaxique qui liste tous les mots en relation avec elle. On

peut alimenter les champs lexicaux depuis le fichier, contenant :

Family : brother sister mother cousin father uncle aunt grandfather grandmother son daughter stepdaughter stepchild

Social : friend enemy rivals know partner friendly buddy bud

Professional : employer employee colleague servant client boss

Couple : husband wife wedding couple lover love wedding married

3 Outils du programme

On code l'application en **Python**, version 3.7, qui est un langage très flexible et agréable à utiliser, particulièrement dans ce contexte. Les librairies les plus importants sont **Spacy** (version 2.1) et **Neuralcoref** (dernière version). Il y a eu des problèmes entre Neuralcoref, qui est une librairie gérant la coréférence, et Spacy. Seule la version 2.1 marche bien avec la dernière version de Neuralcoref. Pour l'installation et l'exploitation du projet, j'en ai profité pour créer un Projet Github, qui contient un README avec une procédure simple d'installation des librairies (utilisant "pip"). Il faut également télécharger un modèle (ici en anglais, de type small).

Attention cependant, l'utilisation de Neuralcoref est très énergivore en ressources et met du temps à s'exécuter sur des récits de taille non négligeable. J'ai voulu utiliser le GPU pour accéléler le traitement de textes, mais cela n'a pas correctement marché sur Windows. Il y a des chances que ce soit efficace sur les systèmes Unix.

4 Position du problème

Pour extraire des relations dans un récit fictif, il faudrait au préalable extraire une liste de tous les personnages. Bien heureusement, Spacy intègre dans son traitement de texte, un outil qui permet de récupérer les termes de type « PERSON » grâce à un système d'entités nommées. Après la récupération et les traitement de la liste des personnages, on peut alors s'occuper des relations existantes entre les personnages. Pour cela, on utilise la méthode « IS-A » , qui est une méthode basée sur les patterns. En effet, à partir du pattern « PERSONNAGE1 [CHAMP LEXCICAL] PERSONNAGE2 » , on peut alors extraire une relation entre PERSONNAGE1 et PERSONNAGE2

par la catégorie syntaxique du champ lexical trouvée. Entre CHAMP LEXICAL et PERSONNAGE2, il y un certain *nombre* de termes qui les séparent potentiellement. La question est alors : à combien d'occurrences on considère le pattern comme correcte ? D'autres problèmes existent comme les doublons de personnages à considérer, des faux postifs sur les personnages, etc.

5 Extraction automatique de personnages

Dans le programme, il y a la possibilité de choisir d'exécuter l'application avec sa liste de personnages (que l'on pourrait annoter dans un fichier à part) ou de la laisser déduire la liste des personnages. Dans le second cas, on utilise Spacy qui liste les termes de type « PERSON ». Puis, on essai concrètement de retirer tous les doublons potentiels. On retirer également tous les nominatifs du type « mrs mme mister ... » pour retirer des ambiguïtés supplémentaire. Un problème survient alors : le système peut croire que les personnages « Alice Lee » et « Alice » sont deux personnages alors que ce sont en réalité les mêmes. C'est pour cela que l'on a adoucit le système d'extraction (avec une expression régulière simple) pour qu'il considère que Alice correspond au personnage Alice Lee (c'est pour cela que j'ai décidé de trier les noms par ordre décroissant de taille, pour avoir le maximum d'informations sur les personnages). Mais cela ne résout pas un autre ambiguïté : « Alice Lee » et « Alice » peuvent être aussi des personnages différents dans le récit. En vérité, c'est moins probable mais évidemment pas impossible et certainement pas négligeable.

Il y a ensuite le problème des coréférences à régler. Il suffit d'utiliser Neuralcoref qui fournit, pour chaque token qu'il considère comme un nom propre, un tableau avec la liste des coréférence (she, he, her them, etc). Il suffit alors de transformer chaque coréférence par le nom associé et de réappliquer le traitement de texte de Spacy. Après plusieurs tests, la librairie ne semble pas très fiable, car elle fait des erreurs non négligeables.

Enfin, il arrive que le système pense qu'une ville ou un pays ou tout autre terme commençant par une lettre capitale, correspond à une entité nommée de type « PERSON ». Il n'y a pas de solution à ce problème, sinon à choisir un modèle plus précis (mais qui prend aussi davantage de temps d'exécution).

6 Extraction de relations

Comme dit précedemment, on utilise la méthode « IS-A ». Le pattern à analyser correspond à celui-ci : « PERSONNAGE1 [CHAMP LEXCICAL] PERSONNAGE2 ». Après réception de la liste des personnages, on boucle sur chaque token du texte. Si un personnage correspond à l'un des personnages (on peut faire la vérification avec des expressions régulières simple, dans un permier temps), on essaie de voir si une relation existe. Pour cela, on cherche alors le prochain champ lexical de PERSONNAGE1. Ensuite, à partir de l'input utilisateur « k » , on décale de k tokens à partir du champ lexical et on recherche le PERSONNAGE2. Si il y en a, on crée une nouvelle relation que l'on ajoute dans un tableau. On vérifiera au préalable si la relation existe déjà (une relation de ce type (A, c, B) est la même que (B, c, A), on a simplement interverti les personnages mais la relation demeure la même). Si la relation est inexistante, on reprend au token où l'on a trouvé le PERSONNAGE1 et on poursuit la recherche jusqu'à la fin du texte.

Ainsi, il est intéressant de faire varier la variable « k » et de comparer les résultats entre eux.

7 Méthodes d'évaluations

On peut évaluer un corpus individuellement en faisant varier « k » et calculer la précision et le rappel en fonction du nombre de résultats corrects obtenus et le cardinal des relations annotées. Ainsi, on peut obtenir soit un graphe rappel-précision, soit un graphe rappel en fonction de k et un graphe précision en fonction de k. Le deuxième est plus pertinent comme on peut le voir sur ces images :

Enfin, on peut évaluer, en se basant sur le deuxième graphe, l'ensemble des corpus. Ceci a été implémenté dans le programme et peut prendre un certain temps d'exécution.

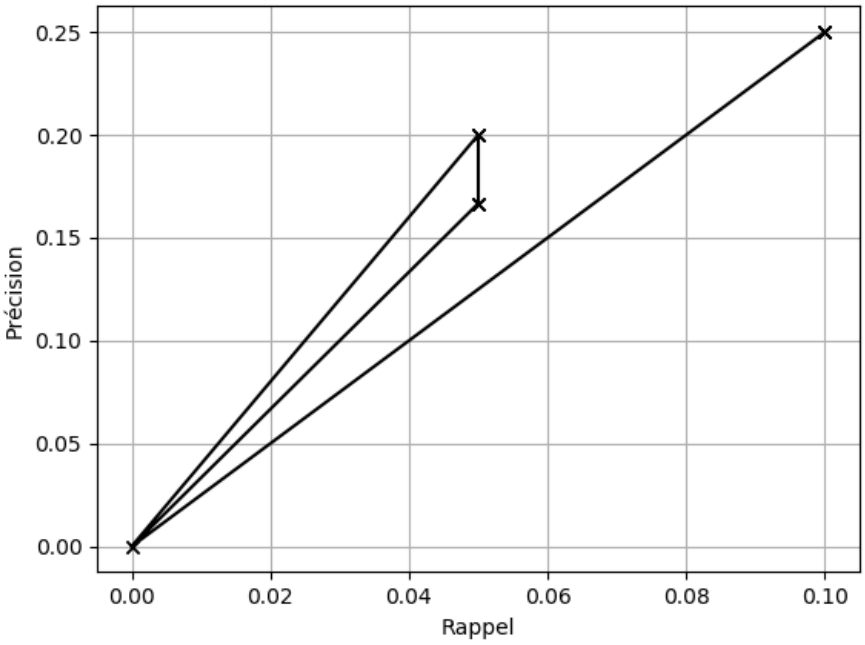

FIGURE 1 – Rappel Précision

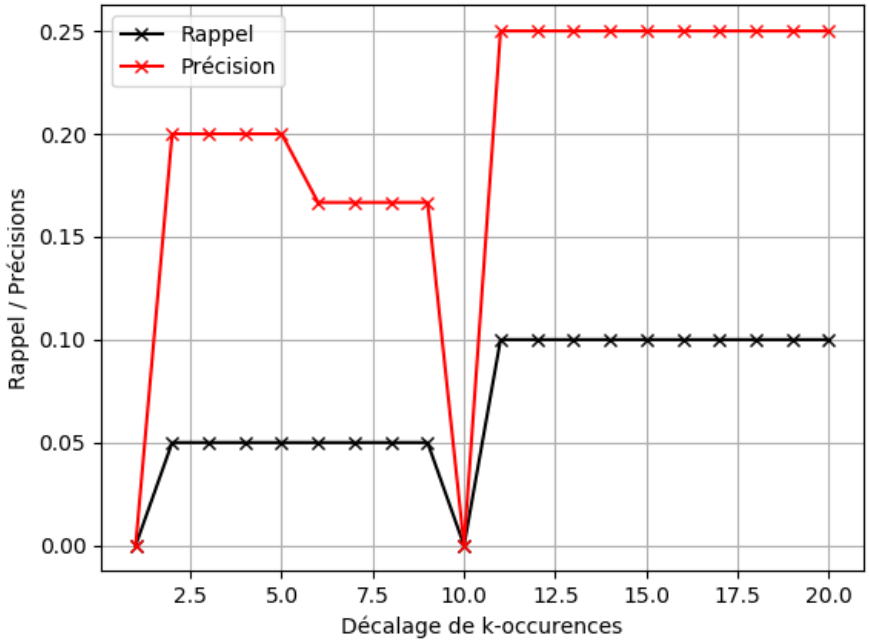

FIGURE 2 – Rappel / Précision en fonction de k

8 Utilisation du programme

On va discuter du mode emploi de l'application. Tout d'abord, on décide dans le programme d'activer ou non la coréférence via la variable **neuralcoref_active**, qui est un booléen. Ensuite, on exécute le bloc de code suivant :

```
name_corpus = "corpus/debug.txt"
name_corpus_annote = "corpus/debug_annote.txt"
characters = run_extration(name_corpus, name_corpus_annote, [], 5,
    True)
"""
on peut remplacer le [] par un tableau listant les personnages. Si
    [] est laiss, on laisse le systme trouver de lui-mme la liste
    des personnages. Le chiffre 5 correspond au dcalage k, et True
    signifie que l'on value le corpus. Tout ceci est mieux expliquer
    au niveau du code

"""
```

Sur Spyder, il est possible d'exécuter une partie du code. Il y a en effet des bout de code que l'on peut exécuter pour tester certaines réponses.

9 Limites du programme

Enfin, de nombreuses limites existent et il m'a semblé nécessaire de les citer. D'un par, la coréférence ne fonctionne pas de manière assez efficace, et prend beaucoup trop de temps à s'exécuter. La coréférence semble être moins efficace que sans dans certaines situations. De plus, l'extraction automatique de personnages dressent une liste potentielle de personnages avec plus de personnes qu'il y en a dans la narration, sans doute à cause des ambiguïtés ou de la pauvreté du modèle utilisé (un r"seau de neurones serait sans doute meilleur).

La méthode de base elle-même est à revoir car on ne considère pas de décalage LIMITE entre PERSONNAGE1 et le CHAMP LEXICAL, seulement entre le CHAMP LEXICAL et PERSONNAGE2 (pour simplifier le code). Et enfin, il n'y a pas d'extraction de relation contextuelle (les non dits par exemple).

Tout ceci n'empêche pas le programme de bien fonctionner sur certains corpus.