Université Paris-Sud

Master AIC - Données Structurées

Apprentissage de Données Structurées : Character Identification in Novels

Julien Louis, Chloé Mercier, Eugène Nélou, Perceval Wajsbürt

Mars 2018

Table des matières

1 Introduction

Le but de ce projet, inspiré par l'article *Identification of Speakers in Novels* de Hua He (University of Maryland), Denilson Barbosa et Grzegorz Kondrak (University of Alberta), est d'identifier les locuteurs dans les dialogues de romans. Cette tâche est difficile à traiter de façon automatique car le locuteur est rarement explicitement nommé. Nous nous servirons des incises du narrateur lorsque cela est possible ; sinon, nous utiliserons le contexte du dialogue, comme les interpellations directes, ou l'alternance des locuteurs dans un dialogue.

Nous utiliserons les dialogues du roman *Pride and Prejudice* de Jane Austen pour entraîner notre modèle.

2 Prétraitement des données

2.1 Correspondance entre le texte d'origine et le dataset

Pour entrainer notre modèle, nous avons besoin de toutes les informations présentes dans le texte d'origine. Cependant, le dataset annoté ne contient pas les mentions narratives décrivant l'énonciateur d'une portion de texte ("he said" est remplacé par "[X]" dans le dataset).

Il nous a donc fallu dans un premier temps effectuer un parsing de l'oeuvre d'origine pour réconstituer le découpage en énonciations et le faire correspondre aux données d'entrainement. Il est à noter également qu'il a fallu effectuer un travail de nettoyage sur le dataset annoté, puisque celui-ci présente des typos (Mrs Phillips au lieu de Mrs Philips) et des énonciations mélangées entre plusieurs personnages.

Le code spécifique au préprocessing se situe dans le fichier *preprocessing.py* de l'archive.

2.2 Dénominations des personnages

Les personnages possèdent souvent plusieurs nom/surnoms parfois difficiles à mettre automatiquement en relation (Lizz et Elizabeth).

Il a donc fallu établir des tables de correspondance entre ces différents noms et les remplacer dans le texte par un représentant de classe (nom principal) pour chacun des personnages. Ce nom principal n'est jamais composé pour faciliter la recherche de mentions de personnages.

2.3 Traitement du texte brut

Pour l'approche Deep Learning décrite plus bas et prenant en entrée un vecteur Bag Of Words, il nous a fallu transformer chaque phrase en token pour obtenir des vecteurs 'indicateurs' de la phrase. Pour réduire la taille du vocabulaire, on a recourt au "stemming" : on réduit chaque mot à sa racine sémantique (ladi pour lady) ce qui nous donne environ 3000 mots distincts, nom principal des personnages compris.

3 Parsing grammatical

3.1 Motivation

On souhaite pouvoir détecter le sujet d'une phrase, des compléments ou encore des appostrophes à certains personnages. Nous avons ainsi choisi d'utiliser le Stanford Parser.

Nous aurions pu obtenir des résultats en observant simplement la nature des mots d'une phrase et leur position, certains termes peuvent être ambigüs (verbe call/nom call) et certains découpages

également : à qui se réfère tel ou tel pronom. Ce couplage peut être résolu grâce à un parseur grammatical probabilistique tel que celui de Stanford.

3.2 Difficultés

Il a notemment été difficile de traiter ainsi les incises du narrateurs dans lesquelles le verbe est en première position car le parseur ne reconnait pas correctement cette structure.

"..." said Kitty

À cette fin, nous avons mis un en place un processus de parsing en deux étapes : nous insérons un faux sujet en tête de phrases et recherchons un complement d'objet au verbe principal, qui se trouvera être le sujet de notre phrase. Une fois de complément trouvé, nous intervertissons la branche factice et le complément, tout en répercutant ces changements sur les différents tableaux représentant la phrase. Enfin, nous supprimons le sujet factice devenu complément, et rééffectuons un parsing sur notre nouvelle phrase mise dans l'ordre.

3.3 Implémentation

Nous effectuons deux types de parsing : un sur les incises du narrateur et un sur les énonciations des personnages. Nous pouvons ainsi traiter toutes les mentions explicites et anaphoriques dont nous parlerons plus loin. Nous avons pris la décision d'exprimer les features découvertes par le parseur sous deux formes

1. un *set* pour les noms de personnages : *{Mr_Bennet, Mrs_Bennet}*
2. une liste de requêtes Prolog : *{related(X,Y,husband), related(Y,X,daughter)}*

4 Relations entre personnages et inférence logique

Le code spécifique au parsing se situe dans le fichier *parsing.py* de l'archive.

4.1 Alias

Certains personnages ont plusieurs alias. Pour éviter la duplication de personnages, nous avons référencé tous les alias d'un personnage dans le texte (en nous appuyant sur le travail de H. He, D. Barbosa et G. Kondrak) et les remplaçons par un identifiant unique en un mot (ex Mr_Bennet).

4.2 Motivation d'une ontologie

Les personnages sont connus a priori, car nous voulons nous concentrer sur le problème de reconnaissance, et ne pas ajouter d'erreurs en amont. Les liens entre personnages (familiaux, amicaux, ...) sont également stockés dans un graphe, et permettent avec l'aide d'ensembles d'alias pour chaque personnage, de reconnaître les mentions anaphoriques faites de ces personnes (eg : *his mother*).

Pour modéliser les règles de description ces relations sociales entre les personnages, nous avons utilisé le langage Prolog.

4.3 Structure

Afin de pouvoir interroger le moteur sur les relations entre deux personnages, nous avons choisi de mettre l'étiquette de cette relation comme paramètre des règles, et de conserver ainsi deux règles uniques :

1. related(X,Y,Relation)

2. status(X,Y,Proprieté)

Dans un but de dynamisme et d'interfaçage aisé avec l'algorithme d'apprentissage, nous avons choisis d'exprimer de la même façon les faits d'entrée et les faits déduits. Par exemple, on souhaite pouvoir déduire des faits `related(kitty, mr_bennet, daughter)` et `related(elizabeth, mrs_bennet, daughter)` le fait `related(kitty, elizabeth, sister)` et inversement des faits `related(kitty, elizabeth, sister)` et `related(mr_bennet, elizabeth, daughter)` le fait `related(kitty, mr_bennet, daughter)`. Ces déductions cycliques nécessitent l'utilisation d'un cache pour éviter les recursion infinies : le module "tabling" de swi-prolog nous permet de réaliser cela.

Bien que nous n'ayons pas recours à l'ajout de faits dynamiquement à la lecture du livre, il nous a paru important d'établir les bases de notre système de telle sorte que cela soit possible à l'avenir, pour des applications telles que celles mentionnées à la fin de l'article, de construction de graphe social.

4.4 Interface avec Python

Pour appeler notre moteur d'inference logique depuis notre algorithme d'apprentissage, nous avons écrit une interface pont entre Python et Prolog, disponible dans le fichier *prolog.py*, qui exécute Prolog comme une routine dans notre algorithme.

```
...
related(X,Y,child):-related(Y,X,parent).
related(X,Y,father):-related(X,Y,parent),status(X,male).
related(X,Y,father):-related(Mother,Y,mother),related(X,Mother,husband).
related(X,Y,mother):-related(X,Y,parent),status(X,female).
related(X,Y,mother):-related(Father,Y,father),related(X,Father,wife).
...
```

FIGURE 1 – Code Prolog

5 Extraction de features

5.1 Mentions explicites

Dans certains cas, le narrateur donne explicitement le locuteur dans une incise. Cette information permet d'identifier à coup sur le locuteur, comme dans l'exemple qui suit :

"Miss Bingley told me," said Jane, "that he never speaks much."

Nous avons identifé les incises à l'aide d'expressions régulières. La principale difficulté est de les interpréter afin de trouver le personnage qui parle. Pour cela, nous nous sommes servi du Part-of-Speech Tagging de Stanford. En récupérant le sujet de l'incise, il est possible d'en extraire le genre, ou la fonction, et si celle si fait partie des relations dérivables par l'ontologie, on peut alors restreindre les personnages possibles, voire en séléctionner un unique.

5.2 Mentions anaphoriques

Les mentions anaphoriques dans un dialogue peuvent permettre d'identifer les personnages qui parlent. Un déterminant possessif permet de limiter les locuteurs possible, voir de les identifier directement puisqu'on connait les relations entre les personnages.

On peut remarquer plusieurs cas selon le possessif :

— Dans une incise, il fait référence à quelqu'un mentionné par le narrateur.
— A la $3^{ème}$ personne, il fait référence à quelqu'un mentionné dans le dialogue.
— A la $2^{ème}$ personne, il fait référence à un autre interlocuteur dans la discussion

— A la $1^{ère}$ personne, il fait référence au locuteur.

Par exemple, l'incise suivante nous permet d'identifier le genre du locuteur et une relation avec un autre personnage, et ainsi de former automatiquement une requête Prolog (dans laquelle on cherche à unifier X avec les personnages potentiels) :

said her husband

```
related(X, Y, husband)
```

On identifie également des propriétés sur le l'autre interlocuteur quand cela est possible, en analysant les différents compléments du verbe principal de la phrase. En rapprochant ces informations de celles de l'énonciateur, on peut restreindre davantage encore la liste des personnages potentiels pour le personnage courant ainsi que celui avec qui il échange.

"..." said the man to his wife

```
related(Y, X, wife), % complement du verbe said "wife"
status(X, male). % pronom possessif "his"
```

5.3 Mentions implicites

Lors d'un dialogue entre deux personnages, les prises de paroles sont souvent alternées. Ainsi, il est très probable que le locuteur d'une phrase *n* soit le même que celui de la phrase *n-2* et *n+2*. Cependant, ce type de règle n'est pas toujours respecté et ne servira que de façon complémentaire dans le choix de l'énonciateur par le modèle.

6 Modèle d'apprentissage sur features manufacturées

6.1 Features

Dans un premier modèle, nous allons extraire pour chaque personnage un ensemble de features au fur et à mesure du parcours des différents dialogues :

— Fréquence de prise de parole du personnage
— Personnage est mentionné dans la phrase énoncée
— Personnage est mentionné par le narrateur
— Personnage est mentionné vocativement dans la phrase précédente
— Genre du locuteur inféré anaphoriquement
— Personnage a déjà pris la parole lors du dialogue actuel
— Personnage a déjà été mentionné lors du dialogue actuel
— Personnage est la cible de la phrase actuelle
— Personnage était la cible de la phrase précédente

Ces features ont été choisies pour rendre possible la modélisation d'un dialogue tel qu'on le conçoit, avec alternance des locuteursou l'interpellation directe des participants d'un dialogue.

6.2 Apprentissage

Nous découpons le roman en bases de train, validation et test. Nous choisissons de regrouper les phrases par dialogues (nous avons besoin de garder des phrases consécutives pour certaines features) et d'effectuer la sélection sur ces dialogues aléatoirement afin d'éviter un biais introduit par la temporalité du roman (les dialogues pourraient ne pas être écrit de la même manière au début et à la fin du roman). Nous entrainons un modèle GradientBoost pour obtenir des performances avec un minimum de paramétrisation.

6.3 Résultats

Les résultats obtenus sont moins bons qu'espérés mais pas étonnant du fait de l'absence de traitement de la langue par le modèle. Les features choisies sont donc moins intéressantes que l'on pensait au premier abord. Notre modèle GradientBoost simple obtient une précision de 20% sur les bases train et validation.

Quand on ajoute la détection du locuteur par mention explicite du narrateur (ex : "Jane said") on gagne en précision. 25% des phrases du corpus sont concernées et détectées par notre traitement. Nour avons 87% de précision sur ces phrases (précision de 22% ramenée au corpus entier). En incorporant ce système de prédiction à notre modèle nous obtenons une précision de 30%. Il semble donc qu'il y ait un fort recouvrement entre les phrases correctement identifiées par les 2 méthodes ce qui implique une grosse importance pour la feature de mention par le narrateur (qui est le point commun des deux systèmes).

7 Modèle basé sur l'ontologie

7.1 Principe

Après avoir effectué le parsing et extrait certaines propriétés sur l'énonciateur et le destinateur au sein d'une uttérance et de l'incise du narrateur, on peut effectuer une requête Prolog en concaténant les différents faits extraits. On obtient ainsi une liste de personnages pour chaque uttérance, identifiés comme énonciateurs ou destinataires.

On fait intervenir les uttérances qui entourent l'uttérance courante et en pondérant les personnages inférés, on effectue un vote majoritaire. Ainsi, en respectant le principe d'alternance, l'énonciateur de l'énonciation n-2 a une pondération positif, et celui de l'énonciation n-1 une pondération négative.

7.2 Résultats

On obtient ainsi 21% de précision en observant seulement l'uttérance courante, et 55% de précision en observant les uttérances qui l'entourent, dont 10% gagnés par ajout des liens sociaux entre les personnages, ce qui montre la pertinence de cette approche.

8 Modèle Deep Learning

Nous avons tenté une approche centrée autour Deep Learning à base de réseaux récurrents (GRU) dans lequel nous injecterons chaque énonciation en vue d'une classification parmis les personnages possibles.

8.1 Modèle retenu

Nous avons choisi la structure décrite dans la Figure 2.

Le texte est découpé en énonciations et chaque énonciation est rapprochée des incises du narrateur qui l'entourent.

Par le parsing de ces incises on extrait les features de genre, nom et rôle (homme, femme) et Prolog permet d'effectuer une estimation des personnages mentionnés dans la scène. À chaque personnage on fait correspondre un embedding, ce qui se traduit par une couche dense après l'entrée one-hot issue de l'ontologie. De plus, contrairement aux autres approches, nous avons souhaité que notre réseau apprenne également du contenu d'énonciation, ce que nous avons effectué grâce à une approche Bag Of Word.

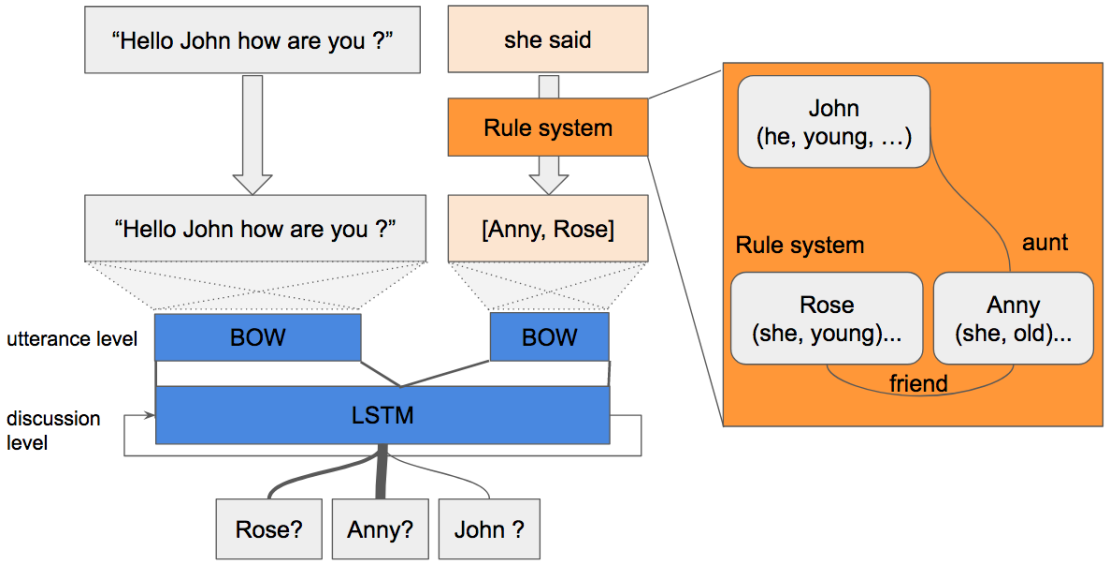

FIGURE 2 – Schéma du modèle Deep Learning

Enfin, comme précisé dans l'article, la propriété d'alternance de prise de parole est une des plus importantes pour la prédiction : on a souhaité pour notre modèle neuronal que cette propriété temporelle soit apprise par le réseau grâce à l'approche récursive. Ainsi, on applique cette couche récurrente sur nos deux entrées concaténées, et on récupère en sortie les prédiction d'énonciateur correspondant à chaque uttérance en entrée.

Enfin, on a choisit comme loss la categorical cross-entropy, adaptée aux problème de classification vers mono-cible. En effet, nous avons ainsi préféré privilegier la majorité des cas ne présentant qu'un énonciateur, plutôt que de traiter les très rares énonciations faisant intervenir plusieurs personnages en même temps.

8.2 Méthodologie de sélection des structures de modèle

Nous avons ainsi décidé d'utiliser Keras et Tensorflow comme frameworks de Deep Learning pour ce modèle.

Comme indiqué dans l'article, nous disposons de connaissances a priori de ce que doit apprendre le réseau :

— prédire le résultat donné par l'ontologie si il est unique,
— alterner les énonciateurs (même sans indice donné par les incises)

Ces connaissances nous ont amenés à tester différentes structures sur des jeux de données générés automatiquement. Par essais successifs, il s'est avéré qu'un réseau utilisant des Gated Reccurrent Unit (GRU) parvenait mieux à apprendre la structure d'alternance qu'un réseau utilisant des Long Short Term Memory units (LSTM). Empiriquement, on constate que le réseau converge plus vite sur les fausses données en plaçant une couche dense après les résultats de l'ontologie, ce qui s'apparente à attribuer un embedding à chaque personnage à cette entrée.

8.3 Apprentissage et résultats

Pour pouvoir batcher les différentes discussions, il nous a fallu définir une taille fixe de discussion et remplir les discussions trop courtes avec des utterances "vides" ne contenant aucun mot et dont le personnage à prédire est un faux personnage.

Pour éviter que ce faux personnage ne devienne prédominant dans nos prédictions, et de façon plus générale pour équilibrer les prises de paroles entre les différents protagonistes, on effectue une pondération de la loss pour chaque exemple.

Malheureusement les résultats sont assez faibles, avec un précision ne dépassant pas 15% pour le set de validation, et pénant à converger pour le set de train. Cela peut indiquer que le problème est trop complexe pour notre modèle, entrainant avec les dimensions de notre structure un phénomène d'underfitting, et notre modèle déjà trop large pour la quantité d'exemples à notre disposition.

Pour les jeux de données de cette taille, on préfère ainsi souvent utiliser des modèles plus "classiques" de machine learning telles que les SVM ou Random Forests.