Identification automatique du discours direct dans un corpus de romans en français

Zhiyuan Qiu

Identification automatique du discours direct dans un corpus de romans en français

Zhiyuan
QIU

Sous la direction de Olivier KRAIF
Tuteur : Agnès TUTIN

Laboratoire : Laboratoire de Linguistique et Didactique des Langues Etrangères et Maternelles - LIDLEM

UFR LLASIC
Département Informatique intégrée en Langues, Lettres et Langage

---

Mémoire de master 2 mention Sciences du langage - 30 crédits

Parcours : Industries de la langue (IDL)

Année universitaire 2021-2022

Identification automatique du discours direct dans un corpus de romans en français

Zhiyuan
QIU

Sous la direction de Olivier KRAIF
Tuteur : Agnès TUTIN

Laboratoire : Laboratoire de Linguistique et Didactique des Langues Etrangères et Maternelles - LIDLEM

UFR LLASIC
Département Informatique intégrée en Langues, Lettres et Langage

---

Mémoire de master 2 mention Sciences du langage - 30 crédits

Parcours : Industries de la langue (IDL)

Année universitaire 2021-2022

Remerciements

Avant de rédiger mon mémoire, je voudrais exprimer ma gratitude à ceux qui m'ont beaucoup appris au cours de mes études, et qui ont fait de ces études un moment important de ma carrière universitaire.

Je tiens tout d’abord à adresser mes remerciements les plus sincères à Monsieur Olivier Kraif pour la confiance qu’il m’a accordée en acceptant d’encadrer ce travail de recherche, pour son soutien tout au long du stage et de la rédaction de ce mémoire, et pour ses nombreux conseils.

Je tiens également à remercier Madame Agnès Tutin pour ses encouragements ainsi que ses conseils durant la rédaction de ce mémoire.

Je remercie également Monsieur Claude Ponton et Monsieur François Portet d’avoir accepté de faire partie de mon jury.

Je tiens également à remercier l'Université Grenoble Alpes et le laboratoire Lidilem pour leur disponibilité et pour avoir toujours transmis leurs connaissances avec passion. Je suis profondément reconnaissant de l'aide et du soutien apportés par vos enseignants et votre équipe académique dans cette période particulière.

Enfin, je souhaiterais très sincèrement remercier ma famille en Chine, mon père, Xiaochu Qiu, ma mère, Dongmei Lin et ma meilleure amie, Yixin Zhu, pour leur soutien et accompagnement tout au long de mes années d’études. Je ne serais pas arrivé là où j’en suis s’ils n’avaient pas été là.

DÉCLARATION ANTI-PLAGIAT

1. Ce travail est le fruit d'un travail personnel et constitue un document original.
2. Je sais que prétendre être l'auteur d'un travail écrit par une autre personne est une pratique sévèrement sanctionnée par la loi.
3. Personne d'autre que moi n'a le droit de faire valoir ce travail, en totalité ou en partie, comme le sien.
4. Les propos repris mot à mot à d'autres auteurs figurent entre guillemets (citations).
5. Les écrits sur lesquels je m'appuie dans ce mémoire sont systématiquement référencés selon un système de renvoi bibliographique clair et précis.

PRENOM : ……Zhiyuan……

NOM : ……Qiu……

DATE : ……02/09/2022……

Sommaire

Introduction

Ce mémoire de recherche a pour but d'explorer les possibilités offertes par le traitement automatique de la langue pour l'identification automatique de discours direct. Ce mémoire fait suite à un stage de recherche effectué dans le cadre du Master 2 en Sciences du langage, parcours Industries de la langue proposé par l'Université Grenoble Alpes. Notre travail de recherche a été encadré par M. Olivier Kraif, Professeur à l'Université Grenoble Alpes et Membre du LIDILEM, ainsi que par Mme Agnès Tutin, Professeur à l'Université Grenoble Alpes et Membre du LIDILEM. Nous avons effectué notre stage au Laboratoire de linguistique et didactique des langues étrangères et maternelles (LIDILEM). Notre travail de recherche s'inscrit dans l'axe 1 du LIDILEM, en lien avec les actions "Phraséologie et pragmatique"[1] et "Phraséologie et corpus romanesques"[2].

Les technologies de l'information déterminent et modifient continuellement la manière dont les gens produisent, vivent et même la façon de penser. Si l'information est immatérielle, elle peut être exprimée dans le langage en tant que support de l'information. En outre, l'avènement d'Internet a fortement influencé l'augmentation de la demande de logiciels permettant de traiter toutes sortes de textes. Cette demande accrue a suscité la recherche de méthodes permettant aux systèmes informatiques de traiter le langage - la focalisation sur le traitement automatique des langues (dorénavant TAL). De manière simple et intuitive, le TAL est l'utilisation de la technologie informatique pour analyser et traiter le langage naturel.

Le discours rapporté est un phénomène linguistique très courant dans les langues humaines, et un discours rapporté est un type de discours qui permet à l'orateur de citer les paroles de quelqu'un d'autre. Cependant, en raison de la richesse et de la complexité du phénomène du discours rapporté, de nombreuses questions méritent d'être approfondies.

L'objectif de notre travail est de développer une méthode d'identification automatique du discours direct dans un corpus romanesque de romans français contemporains (issu du projet PhraseoRom), en utilisant le TAL. La demande d'identification automatique du discours direct représenté concerne le domaine des études littéraires, parce que pour l'étude de la littérature, la détection du discours direct dans la fiction permet de mieux comprendre

[1] https://lidilem.univ-grenoble-alpes.fr/thematiques-recherche/actions-recherche/phraseologie-et-pragmatique
[2] https://lidilem.univ-grenoble-alpes.fr/thematiques-recherche/actions-recherche/phraseologie-et-corpus-romanesques

un élément important de sa structure narrative (Jannidis et al., 2018). De plus, pour les textes informatifs, il est également intéressant de savoir qui a dit quoi. Dans les études linguistiques, il s'agit de comprendre comment fonctionne l'oral représenté, par exemple pour l'étude des phraséologismes pragmatiques ou des motifs spécifiques (Novakova et al., 2019). Il s'agit également d'une exigence pour les études concernant la représentation des personnages, comme l'analyse des sentiments ou les réseaux de personnages (Tu et al., 2019). Schöch et al. (2016) notent également que dans l'histoire des genres littéraires, la détection du discours direct permet d'observer à grande échelle les distributions et les évolutions d'un aspect fondamental et formel du roman. De plus, l'extraction de discours direct permet une analyse plus détaillée du discours du narrateur (Schöch et al., 2016). Étant donné que ces travaux reposent sur le traitement d'une grande quantité de matériel textuel, il est essentiel de disposer d'un moyen d'identification automatique du discours direct. Pour cet objectif, ce mémoire tente de proposer des solutions au problème suivant : comment identifier automatiquement le discours direct dans un corpus romanesque ?

Dans les textes écrits, nous pouvons nous appuyer sur une variété de moyens pour identifier les limites de début et de fin du discours direct. Outre certains traits typographiques, comme le début d'un tiret, nous pouvons également identifier les limites du discours direct à l'aide d'autres moyens, comme les verbes introducteurs, les signes de ponctuation, les pronoms personnels, etc. Ces moyens ne fonctionnent pas de manière isolée ; dans la pratique, ils sont souvent utilisés ensemble pour fonctionner comme des marqueurs de limites du discours direct.

Les objets à identifier automatiquement dans notre travail sont les parties textuelles du discours direct. En fait, l'apparition du discours direct dans les textes écrits est complexe, avec parfois uniquement du discours direct et parfois il y a du discours direct et des séquences introductives de discours direct. Il s'agira donc d'une entreprise complexe et ce, en premier lieu, de par la forme sous laquelle il apparaît n'est pas unique (il est possible d'avoir des séquences introductives de discours direct). Au-delà de cette difficulté, s'ajouteront celles plus générales de la complexité de la syntaxe, et de l'ambiguïté de la langue, qui posent inexorablement un problème dans le traitement automatique des langues.

La première partie de ce mémoire traitera du domaine de la linguistique, en proposant une description de certaines notions fondamentale liées au discours direct. Il fournira également un résumé des formes du discours direct qui apparaissent dans les romans

français. Nous ferons ensuite, dans la Partie II, un bref inventaire des approches possibles qui s'offrent à nous en identification automatique du discours direct.

Après avoir ainsi expliqué notre objet d'identification et énuméré les méthodes offertes par le traitement automatique de la langue pour son identification à partir de corpus, nous expliquerons, dans la troisième partie, les méthodes que nous avons choisi d'utiliser pour l'identification automatique de discours direct. La partie IV fournira ensuite une analyse des résultats produits par les approches que nous avons adoptées pour identification automatique de discours direct.

Partie 1
-
LE DISCOURS DIRECT DANS LES CORPUS ROMANESQUE

Chapitre 1. QU'EST-CE QUE LE DISCOURS DIRECT

Il est facile de voir que le cœur de notre travail concerne les passages d'oral représenté dans le roman français, sous la forme de discours direct. Afin de pouvoir les identifier automatiquement, il nous semble donc nécessaire de définir ce qu'est le discours direct, ainsi que d'en préciser les caractéristiques.

Nous commencerons, dans ce chapitre, par donner certaines notions fondamentales impliquées dans le discours direct.

1. LE DISCOURS RAPPORTE

Lorsqu'il s'agit de discours direct, un autre concept doit être mentionné - le discours rapporté. « *La propension des humains à communiquer, à l'oral comme à l'écrit, les amène régulièrement à rapporter ou à reprendre les propos d'autrui dans une visée communicative* » (Ceccaldi-Hamet, 2020, p.1). Comme le rappelle Monique De Mattia (2000, p.7), « *le discours rapporté constitue une part importante de notre activité de locuteur et intéresse au premier chef la recherche linguistique* ».

Le discours rapporté se définit comme « *l'insertion d'un énoncé source dans un autre énoncé* » (Abeillé et al., 2021, p.2053). Il représente un doublement de l'énonciation. « *Le discours tenu par un locuteur de base contient un discours attribué à un autre dénonciateur (ou parfois au locuteur de base à un autre moment), qui est rapporté par le locuteur premier. Celui-ci se fait en quelque sorte le porte-parole du discours de l'autre locuteur* » (Riegel et al., 1994, p.1009 - 1010).

Ex :

1) *— Quel homme est-ce que ce comte de Monte-Cristo ? demanda Franz à son hôte.*[3]

La représentation du discours rapporté peut être distinguée en trois différentes formes : *discours direct* (voir l'exemple 2), *discours indirect* (voir l'exemple 3) ou *discours indirect libre* (voir l'exemple 4) (Abeillé et al., 2021 ; Riegel et al., 1994).

Ex :

2) *Le chef a dit : « Nous aurons fini ça pour ce soit ! ».*[4]

[3] « Le Comte de Monte-Cristo », Alexandre Dumas, 1889
[4] « La grande grammaire du français », Anne Abeillé et Danièle Godard, p.2053

3) *Le chef a dit que nous aurions fini ça pour ce soit.*[5]

4) *Le chef avait fait ses prévisions. On pouvait finir ça pour ce soit.*[6]

En l'exemple 2, le locuteur présente comme un énoncé indépendant, prononcé par *le chef*. En l'exemple 3, bien qu'il soit le même discours source que l'exemple 2, il n'est pas présenté comme un discours indépendant. Il se trouve dans une complétive en *que*. Enfin, en l'exemple 4, la deuxième phrase est comprise comme le contenu des pensées du *chef,* mais constitue syntaxiquement une phrase distincte.

2. LE DISCOURS INDIRECT

Avant de passer au discours direct, présentons brièvement le discours indirect. Pour le discours indirect, nous avons la définition suivante. *« Il se construit comme une proposition subordonnée, qui est complément d'un verbe principal signifiant « dire » ou « penser » »* (Riegel et al., 1994, p.1012). Le discours indirect perd son indépendance syntaxique et énonciative. Les mots rapportés de cette manière ne peuvent pas être distingués du reste du texte, contrairement au discours direct, où l'auteur doit utiliser les tirets et les guillemets.

Le discours indirect est introduit dans le récit et intégré à celui-ci. Par conséquent, les caractéristiques du discours direct disparaissent. En ce qui concerne les caractéristiques du DI, nous pouvons les résumer comme suit :

- La présence des guillemets et des tirets n'est pas requise.
- Un verbe de parole doit être présent dans le discours indirect (dire, supplier, demander, etc.).

Ex :

5) *Il m'a dit qu'il avait trouvé ce spectacle beau.*[7]

3. LE DISCOURS DIRECT

« Le discours direct constitue apparemment la forme la plus littérale de la représentation du discours d'autrui » (Riegel et al., 1994, p.1010). Ou, en termes plus

[5] « La grande grammaire du français », Anne Abeillé et Danièle Godard, p.2053
[6] « La grande grammaire du français », Anne Abeillé et Danièle Godard, p.2053
[7] Alloprof

simples, le discours direct est un rapport mot à mot de ce que quelqu'un a dit et il est présenté tel quel, comme une citation.

Les caractéristiques du discours direct peuvent être résumées de la manière suivante :

- *« Une ponctuation spécifique qui encadre le passage au discours direct »* (LOG, s.d.). Les deux ponctuations spécifiques généralement utilisées sont le tiret et les guillemets ; tirets devant chaque réplique, guillemets qui s'ouvrent devant la 1ère réplique et qui se ferment après la dernière.
- Il y a des verbes introducteurs (*dit-il ; murmura-t-il…*). Ces verbes peuvent être placés dans des positions variées, soit avant la réplique, soit à la fin de la réplique, soit en incise.
- Les personnes et les temps utilisés : majoritairement, pour le temps du discours sont le présent d'énonciation et le passé composé ; les personnes sont à la première personne (je ; nous) et à la deuxième personne (tu ; vous).

Ex :

6) *Un agneau se désaltérait*
Dans le courant d'une onde pure ;
Un loup survient à jeun, qui cherchait aventure,
Et que la faim en ces lieux attirait.
« Qui te rend si hardi de troubler mon breuvage ?
Dit cet animal plein de rage :
Tu seras châtié de ta témérité.
– Sire, répond l'agneau, que Votre Majesté
Ne se mette pas en colère (…) »[8]

Nous devons ici présenter brièvement une autre notion, à savoir l'incise （La phrase incise). « *La phrase incise accompagne généralement les citations et les dialogues. Elle indique qui prend la parole dans un discours rapporté direct* » (Alloprof, s.d.).

La phrase incise contient un verbe de parole (*dire, répondre, répliquer, chuchoter, etc.*) et, en même temps, la précision concernant l'interlocuteur. En général, la phrase incise « *se détache du reste de la phrase par une ou des virgules* » (Alloprof, s.d.) (Comme dans l'exemple 8 et 9 ci-dessous), bien que d'autres signes de ponctuation puissent être utilisés. (Comme dans l'exemple 7 ci-dessous)

Ex :

[8] « Le Loup et l'Agneau », Jean de La Fontaine, 1668

7) *— Pourquoi auraient-ils décidé de briser un silence de dix siècles? demanda le consort.*[9]

8) *« Nous ne prendrons pas le risque de manquer le passage des visiteurs célestes », déclara la reine Osfoët.* [10]

9) *« Nous ne pensions pas que quelqu'un viendrait nous chercher sur cet îlot, poursuivit-il. Nous attendions d'être remis de notre voyage pour repartir. Je... »* [11]

L'image ci-dessous résume les caractères spécifiques de chaque type de discours rapporté.

Caractéristiques grammaticales	**Discours direct**	**Discours indirect**
Verbes de paroles / pensées	✘●	✘
Dépendance syntaxique		✘
Frontière marquée (tirets, majuscules, etc.)	✘	
Modalité (assertion, exclamation, interrogation, etc.)	✘	C'est la principale qui la conditionne.
Marque de l'oralité (interjection, jargon, etc.)	✘	
Transposition	✘	✘

Figure 1. Les caractères spécifiques de chaque type de discours rapporté (etudes-litteraires, s.d)[12]

4. *LE DISCOURS INDIRECT LIBRE*

Contrairement aux deux types mentionnés ci-dessus (le discours direct et le discours indirect), le discours indirect libre est présenté « *comme combinant des caractéristiques du discours direct et du discours indirect* » (Abeillé et al., 2021, p.2063). Il se distingue de ces deux formes de discours rapporté (le discours direct et le discours indirect) par l'absence de marqueurs spécifiques (Abeillé et al., 2021 ; Riegel et al., 1994) :

- Pas subordonné à un verbe de parole
- Accompagné d'aucune indication typographique spécifique (guillemets etc.)

Ex :

10) *Ils mouraient avec leurs poux dans les cheveux et dès qu'ils étaient morts le père disait, c'est bien connu, les poux quittent les enfants morts, il faut l'enterrer tout de suite sans*

[9] « Bordage »
[10] « Bordage »
[11] « Bordage »
[12] La croix marque un caractère obligatoirement présent, tandis que le rond et la croix indiquent un caractère non obligatoire, mais parfois présent.

ça on va être envahi, et la mère, attends que je le regarde, et le père, que deviendrons-nous si les poux se mettent dans la paillote de la case ? Et il prenait l'enfant mort et l'enterrait encore chaud, dans la boue, sous la case [13]

La partie soulignée est le discours indirect, la partie restante est le discours direct. En raison du verbe introducteur ainsi que les marques de ponctuation (guillemets ou tirets) sont supprimés dans le discours indirect libre, il n'est pas toujours facile de distinguer le discours direct et le discours indirect.

5. CONCLUSION

Comme le mentionne Joël July (2016) dans son article, puisque le discours indirect libre *« n'utilise pas les marqueurs (lexicaux et typographiques) qui répartissent la parole entre les différents personnages »*, le discours indirect libre est plus susceptible de créer des confusions (il est difficile de distinguer les deux énoncés). Le discours direct est donc relativement facile à reconnaître. Pour notre travail, l'identification automatique de discours direct, nous devrons examiner le corpus que nous utilisons ; si c'est le discours direct, notre travail d'identification automatique de discours direct sera plus facile et nous pourrons nous concentrer sur la ponctuation et le verbe introducteur. S'il s'agit du discours indirect libre, nous pouvons avoir besoin d'autres informations pour nous aider à l'identifier.

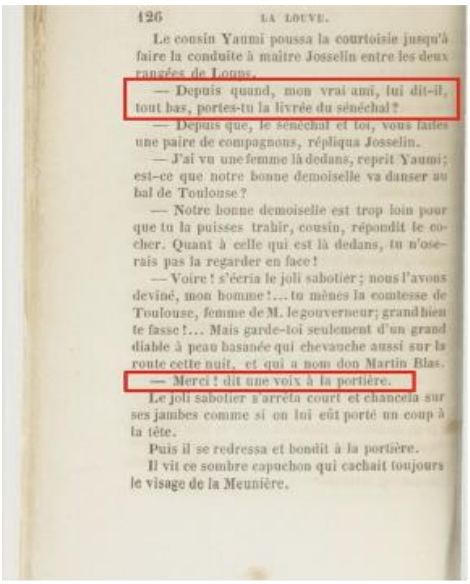
[13] « Un barrage contre le Pacifique », Marguerite Duras, 1950

Chapitre 2. LES TYPES DE PRESENCE

Dans la première partie, nous avons présenté les notions de base du discours direct, ce qui sera utile pour notre recherche. Cependant, nous constatons que l'utilisation du discours direct peut varier, par exemple, dans les paroles prononcées, écrites ou inventées. Les stratégies typographiques du discours direct peuvent être différentes selon le type de texte : par exemple, le retrait à la ligne avec tiret (le roman), les italiques, le retrait en paragraphe spécial (le texte scientifique). Nous devons donc identifier automatiquement le genre textuel que nous cherchons à identifier afin de clarifier la forme exacte sous laquelle le discours direct se produit. Le discours direct que nous devons identifier automatiquement dans ce mémoire provient de différents romans français. Dans ce chapitre, nous présenterons donc une classification des formes sous lesquelles le discours direct existe dans le roman français.

1. LE DISCOURS DIRECT

Pour l'étude du discours direct dans les romans français, il faut mentionner le facteur de la typographie. Dans les travaux de Schöch et al. (2016) et Kurfali et al. (2020), on peut constater que dans la tradition typographique française, le discours direct et le dialogue ne sont généralement pas marqués par des guillemets ouvrants et fermants (figure 1). Au contraire, un tiret long (—) indique généralement le début du discours direct, tandis que la fin n'est pas marquée (Kurfali et al., 2020 ; Schöch et al., 2016).



126 LA LOUVE.

Le cousin Yaumi poussa la courtoisie jusqu'à faire la conduite à maître Josselin entre les deux rangées de Loups.

— Depuis quand, mon vrai ami, lui dit-il, tout bas, portes-tu la livrée du sénéchal?

— Depuis que, le sénéchal et toi, vous faites une paire de compagnons, répliqua Josselin.

— J'ai vu une femme là dedans, reprit Yaumi; est-ce que notre bonne demoiselle va danser au bal de Toulouse?

— Notre bonne demoiselle est trop loin pour que tu la puisses trahir, cousin, répondit le cocher. Quant à celle qui est là dedans, tu n'oserais pas la regarder en face!

— Voire! s'écria le joli sabotier; nous l'avons deviné, mon homme!... tu mènes la comtesse de Toulouse, femme de M. le gouverneur; grand bien te fasse!... Mais garde-toi seulement d'un grand diable à peau basanée qui chevauche aussi sur la route cette nuit, et qui a nom don Martin Blas.

— Merci! dit une voix à la portière.

Le joli sabotier s'arrêta court et chancela sur ses jambes comme si on lui eût porté un coup à la tête.

Puis il se redressa et bondit à la portière.

Il vit ce sombre capuchon qui cachait toujours le visage de la Meunière.

Figure 2. « *La Louve* », 1857, p. 126[14]

D'après les travaux des chercheurs, il existe un cas relativement simple du discours direct dans le roman français, où la partie qui se produit est entièrement en discours direct, c'est-à-dire que, le passage ne contient que des discours directs et aucune autre phrase. Cela se produit généralement dans les passages de dialogue (voir par exemple Byszuk et al., 2020 ; Schöch et al., 2016 ; Sini et al., 2018).

Ex :

11) *– D'où viens-tu encore, gredin?*
Vous êtes bien curieux, sans yeux... [15]

En outre, étant donné que les romans n'utilisent pas tous la même typographie et qu'il n'existe pas de norme typographique fixe qui limite la publication des romans française, nous constatons que la présence du discours direct apparaît parfois entre les guillemets français (voir par exemple Byszuk et al., 2020 ; Pouliquen et al., 2007 ; Sini et al., 2018).

Ex :

12) *« Votre cousin Barino n'est plus si riche qu'autrefois. Il a même consenti à me vendre sa villa, ma chère Gilberte, et je vous l'offre aujourd'hui. Elle est à vous. »*[16]

[14] Source :http://gallica.bnf.fr/ark:/12148/bpt6k6366934b
[15] « Les Mystères de Paris », Eugène Sue, 1842
[16] « Fleur de pavé », Pascal Lainé,1996

2. LE DISCOURS DIRECT AVEC INTRODUCTEURS DE DISCOURS DIRECT

Outre le cas plus simple mentionné ci-dessus, où la phrase ne contient que de discours direct, nous avons un autre cas de figure connue sous le nom de cas mixte : discours direct + introducteurs de discours direct. Ce cas est relativement complexe, car les parties qui apparaissent ne sont pas toutes intégrées dans le discours direct (voir par exemple Byszuk et al., 2020 ; Schöch et al., 2016 ; Sini et al., 2018).

Ce cas mixte, à son tour, peut être divisé selon la complexité de la structure comme suit :

- Cas simple : **Des discours directs + d'autres syntagmes.**

Le discours direct se termine après que le locuteur ait été indiqué via un verbe introducteur (comme *dire, ajouter,* etc.) ou le discours direct apparaît après que le locuteur a été introduit.

Ici, le verbe introducteur peut être placé avant ou après le discours direct. Dans le cas du discours direct avec un tiret long, le verbe introducteur devrait normalement suivre le discours direct, comme dans l'exemple 13 ci-dessous, mais dans le cas du discours direct avec les guillemets français, le verbe introducteur peut apparaitre soit avant le discours direct (comme dans l'exemple 15 ci-dessous), soit après le discours direct (comme dans l'exemple 14 ci-dessous).

Ex :

13) *– Merci !* ***dit une voix à la portière.***[17]

14) ***Elle me murmura :*** *« Reste digne de lui. »*[18]

15) *« D'accord, comme Rocky,* ***a-t-elle ajouté en riant.*** *»*[19]

- Cas compliqué : **Des discours directs + incise (introducteurs de discours direct) + des discours directs**

Contrairement au cas mixte simple mentionné ci-dessus, il est facile de voir que notre phrase ne se termine pas après l'incise, et que la partie incise est suivie de la partie du discours directe. Dans un tel cas, le nombre d'éléments d'incises est indéterminé, il peut n'y en avoir que la partie d'introduction du discours, comme dans l'exemple 17 et 18 ci-dessous,

[17] « La Louve », Paul Féval, 1857
[18] « Fleur de pavé », Pascal Lainé,1996
[19] « Registre des morts », Patricia Cornwell, 2008

ou la partie d'introduction du discours avec une partie descriptive, indiquant la manière, comme dans l'exemple 16 et 19 ci-dessous (voir par exemple Byszuk et al., 2020 ; Schöch et al., 2016 ; Sini et al., 2018).

16) *– Depuis quand, mon vrai ami, **lui dit-il, tout bas,** portes-tu la livrée du sénéchal ?*[20]

17) *– Levez-vous, **reprit le professeur,** et dites-moi votre nom ...*[21]

18) *« Je conçois, **s'excusait-il**, que cette vilaine bâtisse perdue dans les bois ne présente guère d'attrait pour une jeune mariée. Mais le candidat que je suis aux élections de septembre ne peut se dispenser de visiter sa future circonscription. »* [22]

19) *« Je ne t'apprends rien.*
*En revanche, ça nous délivre un grand nombre d'informations cruciales, **déclame Sebastian, l'index pointé en direction de l'image en suspension.***
Cruciales en matière de prospective et de cosmologie. » [23]

3. DIFFICULTES D'IDENTIFICATION

Après avoir résumé les occurrences possibles du discours direct dans les romans français, nous avons constaté que le discours direct n'apparaît pas toujours de manière unique, mais qu'il est parfois accompagné de certaines parties d'introducteurs de discours direct. Avec la présence de ces introducteurs de discours direct, il sera sans doute plus difficile pour nous d'identifier automatiquement le discours direct. Nous présenterons les difficultés que nous pouvons rencontrer lors de l'identification automatique du discours direct, ce qui aidera la partie pratique de notre travail par la suite.

3.1. SEULEMENT LE DISCOURS DIRECT

Il est facile de voir que l'identification du discours direct est relativement simple pour les passages qui ne contiennent que le discours direct ou si le discours direct est précédé d'un marqueur symbolique non ambigu, tel qu'un tiret long (—) ou des guillemets français (« … »), etc. Le contenu complet de la conversation est le discours direct à identifier (des passages de dialogue). Il suffit de trouver des marques de début claires pour les identifier. Dans ce cas, nous considérons que les marques de début sont non ambiguës et intuitives, c'est-à-dire que nous pouvons utiliser des symboles typographiques tels que le tiret long (—) ou les guillemets français (« … »), comme marques de début.

[20] « La Louve », Paul Féval, 1857
[21] « Madame Bovary », Gustave Flaubert, 1856
[22] « Fleur de pavé », Pascal Lainé,1996
[23] « Involution », Johan Heliot, 2014

Ainsi, pour l'identification automatique de cas ne contenant que du discours direct, nous pensons que la difficulté réside dans la détermination de la ponctuation impliquée dans l'étiquette de début.

3.2. LE DISCOURS DIRECT AVEC INTRODUCTERUS DE DISCOURS DIRECT

Dans les exemples mentionnés au chapitre 2, les parties soulignées sont des discours directs. Quant aux parties en gras, ces sont des incises, que nous pouvons également appeler les parties de l'introducteurs de discours direct. Ainsi, contrairement à la structure habituelle qui ne contient que du discours direct, certaines phrases contiennent également des parties de l'introducteurs de discours direct, et tout est placé après un tiret long (—) ou entre les guillemets français (« … »). Le début et la fin du discours direct ne sont pas clairement marqués formellement, ni distingués par des sauts de ligne, mais simplement par une séquence de phrases, ce qui rend difficile de dire quelle partie est du discours direct et quelle partie concerne du discours indirect sans une identification minutieuse.

À partir de ce phénomène linguistique, les questions suivantes viennent à l'esprit : si les parties du texte qui sont les discours directs sont avec les introducteurs de discours direct, par quel moyen le lecteur reconnaît-il les limites de début et de fin du discours direct ? Un ordinateur serait-il capable de faire le bon choix si on lui permettait de l'identifier ?

Par conséquent, nous comprenons que dans le cas où le discours direct et l'introducteur du discours direct sont présents en même temps, la difficulté de l'identification automatique du discours direct réside dans la manière de réussir à identifier la partie du discours direct ajoutée après l'introducteur du discours direct. En d'autres termes, comment déterminer les limites de début et de fin de l'introducteur du discours direct. Dans ce cas, se baser uniquement sur la ponctuation (par exemple, tiret long, les guillemets français, etc.) n'est pas un moyen précis de déterminer et d'identifier les discours directs.

Partie 2
-
ETAT DE L'ART DES METHODES D'IDENTIFICATION AUTOMATIQUE

Chapitre 3. APERÇU DU PROCESSUS D'IDENTIFICATION AUTOMATIQUE DU DISCOURS DIRECT

Dans ce chapitre, nous donnerons un aperçu des méthodes actuelles pour l'identification automatique de discours direct en TAL. D'après les études publiées, il existe deux grands types d'approches applicables à l'identification automatique du discours direct. Il s'agit des approches symboliques et neuronales (impliquant un apprentissage automatique). Une présentation plus détaillée de ces deux approches sera développée dans le chapitre suivant.

1. PRETRAITEMENT

Bien que les approches adoptées soient différentes, nous trouvons un dénominateur commun dans le fait que toutes deux nécessitent un travail de prétraitement du texte. Nous donnerons d'abord une brève description de ces prétraitements, en guise d'introduction.

Avant de procéder à l'identification automatique, il est nécessaire de prétraiter les données afin d'identifier certaines caractéristiques utiles pour la suite. Ces prétraitements peuvent être décomposés en la suite d'opérations suivantes :

1.1. Segmentation de la base de données/document analysé

Il s'agit du découpage du document en paragraphes, puis du découpage des paragraphes en phrases. La décision de poursuivre ou non la segmentation des phrases en tokens dépend de l'approche utilisée (Jannidis et al., 2018 ; Liang et al., 2010 ; Poulard et al., 2008).

1.2. Analyse linguistique et sémantique

Dans cette section, les chercheurs font essentiellement un étiquetage grammatical (verbe, nom, etc.) des mots de chaque phrase. Il n'y a pas d'exigence uniforme quant à l'analyse syntaxique (sujet, prédicat, etc.) de chaque phrase. Schöch et al. (2016) et Fabien et al. (2008) ont également effectué une lemmatisation dans leurs travaux.

Pour les travaux de segmentation, d'étiquetage grammatical et d'analyse syntaxique, il existe un certain nombre de logiciels. Par exemple, *TreeTagger* est utilisé par Schöch et al. (2016) et Poulard et al. (2008), *Stanford dependencies*[24] et *Stanford Parser* [25] sont utilisés par Pareti et al. (2013), Schöch et al. (2016), ainsi que *Stanford CoreNLP-Tokenizer* et *Sentence-*

[24] https://nlp.stanford.edu/software/stanford-dependencies.html

[25] https://nlp.stanford.edu/software/lex-parser.html

Splitter. *OpenNLP Tokenizer*[26], *OpenNLP SentenceDectector*[27] et *RFTagger*[28] sont utilisés par Tu et al. (2019). Outre les logiciels mentionnés ci-dessus, il existe bien sûr un certain nombre d'autres options. Le logiciel le plus approprié peut être choisi en fonction des exigences de la tâche. Il faut noter que ces prétraitements sont plutôt spécifiques aux méthodes symboliques.

1.3. Pour l'approche d'apprentissage automatique

Dans leur approche d'apprentissage automatique destinées à l'identification automatique du discours direct, Jannidis et al. (2018) ont choisi de supprimer tous les guillemets lors du prétraitement des données, dans le but d'éviter que le modèle entraîné se fonde uniquement sur l'indice formel des guillemets. En outre, Kurfali et al. (2020) soulignent dans leur article que la suppression des signes de ponctuation, par exemple les guillemets simples et doubles (guillemets simples : *'...'*, guillemets doubles : *"... ")*, n'est pas seulement destinée à permettre au modèle d'apprendre les caractéristiques linguistiques du discours direct, mais aussi à éviter les difficultés de l'identification automatique de discours direct dues aux différences de conventions typographiques entre les périodes ou les langues.

2. CONSTITUTION D'UNE REFERENCE

Après avoir prétraité les données, il est important de constituer un corpus de référence qui permettra à la fois de développer les règles ou d'entrainer les modèles, et d'évaluer le système. Dans un tel corpus, il faut classer et étiqueter le texte prétraité afin de distinguer les données qui appartiennent au discours direct de celles qui n'y appartiennent pas.

Il est possible de diviser simplement le texte en deux catégories comme le font Schöch et al. (2016) : contenant du discours direct et ne contenant pas de discours direct ; Tu et al. (2019) utilisent une division du texte en deux catégories : narrative et non-narrative (le discours direct est dans la catégorie non-narrative) [29]. Il est également possible de diviser le texte en trois catégories comme dans Sini et al. (2018) : le groupe discours direct, le groupe narration et le groupe mixte (c'est-à-dire discours direct + narration) ; on peut aussi mentionner les trois catégories utilisées dans Pouliquen et al. (2007), pour l'identification de citation : *« speaker*

[26] https://opennlp.apache.org/docs/1.8.2/apidocs/opennlp- tools/opennlp/tools/tokenize/Tokenizer.html

[27]https://opennlp.apache.org/docs/1.8.2/apidocs/opennlp-tools/opennlp/tools/sentdetect/package-summary.html

[28] https://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/

[29] Car il est peu probable qu'une section narrative contienne du discours direct

name», *«reporting verb»* et *«quotation»*[30], etc. Ce à quoi il faut faire attention, c'est de faire la distinction entre l'identification des citations dans la presse et l'identification de dialogue dans les romans. Ce n'est pas exactement la même chose, mais nous pouvons nous inspirer des méthodes qu'ils utilisent et les modifier pour notre travail.

Après avoir identifié ces différentes catégories d'annotation, nous devons marquer les données pour distinguer les parties qui sont du discours direct et celles qui n'en sont pas, en fonction du format du texte et aussi de l'approche utilisée.

2.1. Annotation des phrases et des tokens en XML

Si un format XML basé sur une segmentation des phrases est utilisé, nous pouvons alors étiqueter chaque phrase. La méthode de marquage utilisée par Barnas (2017) en est un exemple. Si une phrase contient une partie de discours direct, alors nous ajoutons une étiquette *« type=dd »*, à cette phrase.

```
<s type="dd"  id="s184">— Ah, s'il vous plaît, n'en parlons plus jamais !
<s type="dd"  id="s186">— Bien, mon ami ! </s><s type=""  id="s187">répon
```

Figure 3. Exemple de la méthode de marquage utilisée par Barnas

Si le format utilisé est basé sur une segmentation des tokens, nous pouvons choisir de marquer chaque token. Il s'agit de la méthode d'annotation que nous utilisons dans notre travail. Nous marquons tous les tokens qui font partie du discours direct avec *« type=dd »*. Ceux qui ne font pas partie du discours direct peuvent être soit non marqués, soit marqués par une autre étiquette.

```
<s id="s25">
    <t id="t436" num="1" l="—" c="PUNCT" e="" f="" type="dd">—</t>
    <t id="t437" num="2" l="aller" c="VERB" e="" f="Mood=Imp|Number=Plur|Person=2|Tense=Pres|VerbForm=Fin" type="dd">Allez</t>
    <t id="t438" num="3" l="," c="PUNCT" e="" f="" type="dd">,</t>
    <t id="t439" num="4" l="Joseph" c="PROPN" e="" f="" type="dd">Joseph</t>
    <t id="t440" num="5" l="!" c="PUNCT" e="" f="" type="dd">!</t>
</s>
```

Figure 4. Exemple de la méthode de marquage utilisée dans notre travail

2.2. Autres formats d'annotation

Nous nous concentrons ici sur l'étiquetage du texte dans les approches d'apprentissage automatique.

[30] Le discours direct est dans la catégorie *« quotation »*

- Une méthode de marquage relativement simple est celle de Byszuk et al. (2020), qui utilisent seulement deux étiquettes, “I” et “O”, dans le texte pour le marquage ; l’étiquette “I” pour les tokens appartenant au discours direct et l’étiquette “O” pour les tokens qui n'appartiennent pas au discours direct. L'annotation est telle que présentée dans l'exemple ci-dessous. (Les annotations sont exprimées au format JSON)

Ex :

—	Hé	bé	!	dit	la	prof	.[31]
I	I	I	I	O	O	O	O

- Kurfali et al. (2020) et Pareti et al (2013) utilisent la même méthode, mais selon le schéma « BIO », avec une étiquette “B” supplémentaire par rapport à Byszuk et al. (2020). Ici, l’étiquette “B” désigne le premier token du segment de discours direct, l’étiquette “I” désigne les tokens qui appartiennent au discours direct (le reste sauf le premier token) et l’étiquette “O” désigne les tokens qui ne font pas partie du discours direct.

Ex :

—	Hé	bé	!	dit	la	prof	.[32]
B	I	I	I	O	O	O	O

- Une utilisation plus détaillée des étiquettes se trouve dans les travaux de Jia et al. (2019). Ils ont utilisé un total de quatre étiquettes pour le marquage, à savoir “BIEO”. L’étiquette “E” correspond ici au dernier token du segment de discours direct.

Ex :

—	Hé	bé	!	dit	la	prof	.[33]
B	I	I	E	O	O	O	O

Après le prétraitement, et l'annotation des données de référence, nous passons à la partie de la reconnaissance automatique de discours direct. Deux familles de méthodes sont aujourd’hui couramment utilisées, les approches symboliques et les approches neuronales.

[31] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[32] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[33] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

Nous donnerons une explication détaillée de ces deux méthodes dans le chapitre suivant pour voir exactement comment elles permettent la reconnaissance automatique de discours direct.

Chapitre 4. PRINCIPALES APPROCHES

Après le traitement de toutes les données, nous arrivons à la partie la plus importante, c'est-à-dire les approches utilisées pour parvenir à l'identification automatique de discours direct. Dans ce chapitre, nous commencerons par présenter en détail deux familles de méthodes qui sont aujourd'hui couramment utilisées. Ensuite, nous donnerons des exemples des erreurs les plus courantes qui se produisent dans la reconnaissance automatique du discours direct.

1. APPROCHES SYMBOLIQUES

Ces méthodes se basent particulièrement sur des règles définies par des experts en langage naturel, généralement des linguistes, et exprimées dans un ou plusieurs formalismes spécialisés (des règles, des expressions régulières, etc.), ou encodées directement dans un langage de programmation commun. Du point de vue de l'histoire du TAL, ce sont les approches les plus anciennes.

La tâche de définition des règles peut s'avérer très laborieuse, car les experts étudient souvent le corpus au cas par cas. En partant des règles les plus générales, ils doivent souvent définir des règles très spécifiques afin de couvrir le plus grand nombre possible de variations de la langue. Cela constitue un long processus de lecture, de relecture, d'analyse et de réflexion.

Pour la rédaction de règles de travail pour la reconnaissance automatique du discours direct, la première chose à déterminer après le prétraitement est de savoir quels sont les signes de ponctuation et les indices linguistiques associés au discours direct, dans le corpus. Les ponctuations peuvent être utilisées comme l'une des principales sources de règles pour l'identification automatique des discours directs.

Barnas (2017) et Sini et al. (2018) nous le rappellent en termes de règles de ponctuation. Outre la section du discours direct commune commençant par un tiret long ou tiret cadratin (—), il convient de noter l'importance de certains autres symboles, tels que les deux points (:), les guillemets ("…" ou « … »), les points de suspension (...) etc. En effet, même dans un corpus romanesque, tous les discours directs n'impliquent pas un tiret long (—) comme le montrent les deux exemples suivants.

Ex :

20) *« **Vous êtes hors de danger, vous m'entendez ? Ça prendra le temps que ça prendra, mais vous êtes sur la bonne voie. Je vais faire prévenir votre famille. Madame Mailloux... Essayez d'ouvrir les yeux ou de serrer ma main, même un peu. Vous m'entendez ?** »*[34]

21) *Le Commandant m'a regardée de ses yeux candides de petit garçon. Il a dit :* ***Ah, oui, j'ai lu les revues, c'est pour cela qu'on faisait l'article, n'est-ce pas ? Mais voyez les statistiques, ma chère.***[35]

En outre, les éléments qui suivent la ponctuation pertinente méritent une attention particulière. Par exemple, le caractère espace joue un rôle précis dans les séquences suivantes, toutes tirées de notre corpus romanesque :

- Les phrases commençant par le tiret long (—) ou le guillemet («) (le guillemet ouvrant est ici accompagné d'un guillemet fermant) +l'espace + la première lettre en majuscule, font généralement partie d'un discours direct (cf. exemples 22 et 23 ci-dessous).

- Les phrases commençant par le tiret long (—) ou le guillemet («) (le guillemet ouvrant ici peut être seul) + l'espace + la première lettre en majuscule + certains verbes introducteurs + ..., sont aussi généralement du discours direct (cf. exemples 24 et 25).

- La partie qui suit l'espace après un verbe introducteur suivi de deux points est généralement du discours direct (cf. exemple 26)

Ex :

22) ***– D'où viens-tu encore, gredin?***[36]

23) *« **Vous êtes hors de danger, vous m'entendez ? Ça prendra le temps que ça prendra, mais vous êtes sur la bonne voie. Je vais faire prévenir votre famille. Madame Mailloux... Essayez d'ouvrir les yeux ou de serrer ma main, même un peu. Vous m'entendez ?** »*[37]

24) *— **Puisque t'en es si sûr…**, répondit Jeannot dans un sourire.*[38]

25) *« **Adrienne**..., murmura-t-elle.*[39]

---

[34] « Revenir de loin », Marie Laberge, 2010
[35] « La servante écarlate », Margaret Atwood, 1985
[36] « Les Mystères de Paris », Eugène Sue, 1842
[37] « Revenir de loin », Marie Laberge, 2010
[38] « Fleur de pavé », Pascal Lainé,1996
[39] « Fleur de pavé », Pascal Lainé,1996

26) *Le Commandant m'a regardée de ses yeux candides de petit garçon. Il a dit :* ***Ah, oui, j'ai lu les revues, c'est pour cela qu'on faisait l'article, n'est-ce pas ? Mais voyez les statistiques, ma chère***.[40]

Par ailleurs, nous avons constaté que la plupart des méthodes s'appuient sur l'observation des *verba dicendi* (reporting verbs). Les recherches ont montré que certains verbes utilisés sont souvent associés au discours direct (Pouliquen et al., 2007 ; Liang et al., 2010 ; Poulard et al., 2008 ; Sini et al., 2019 ; Tu et al., 2019). Par exemple, *souligner*, *annoncer*, *dire*, etc. En outre, du fait que certains verbes spécifiques sont souvent utilisés en cas de discours direct, le temps du verbe peut également aider à déterminer s'il s'agit d'un discours direct (Tu et al., 2019). Par exemple, si le verbe est conjugué à la troisième personne et si le temps est passé, il ne s'agit généralement pas d'une partie de discours direct. Il est ainsi utile de prendre en compte les verbes dans les règles d'identification (cf. exemple 27).

Sini et al. (2018) font les recommandations suivantes pour l'identification du discours directs dans leur travail pour les cas où le verbe est détecté comme la troisième personne du singulier (ce qui, selon le texte, est satisfait dans 97% des cas) :

- Si le sujet est à gauche, c'est-à-dire avant le verbe, il s'agit d'une amorce.

Ex :

27) *Puis, l'ayant considéré quelques minutes d'un œil amoureux et tout humide,* ***elle*** *dit vivement :*
— Ah ! tenez, n'en parlons plus… Où sont les chevaux ? Retournons.[41]

- Si le sujet est à droite, c'est-à-dire après le verbe, il s'agit d'une incise. Dans ce cas, il est important d'en établir l'extension, les incises pouvant être courtes (cf. exemple 28) ou relativement longue (cf. exemple 29).

Ex :

28) *– Levez-vous,* ***reprit le professeur****, et dites-moi votre nom…*[42]

29) *– Débarrassez-vous donc de votre casque,* ***dit le professeur, qui était un homme d'esprit***.[43]

De même, les pronoms personnels sont également importants à noter dans les règles d'identification, car il est parfois possible de juger s'il s'agit d'un discours direct au moyen

[40] « La servante écarlate », Margaret Atwood, 1985
[41] « Madame Bovary », Gustave Flaubert, 1856
[42] « Madame Bovary », Gustave Flaubert, 1856
[43] « Madame Bovary », Gustave Flaubert, 1856

des pronoms personnels (ici, c'est à l'intérieur du discours, et non à sa frontière). Par exemple, les pronoms à la première et à la deuxième personne impliquent un discours direct dans la plupart des cas. Il convient toutefois de noter que la conjugaison des verbes et les pronoms personnels ne peuvent pas être utilisés comme règles d'identification à eux seuls ; ils doivent être utilisés en conjonction avec d'autres contraintes. Par exemple, lorsqu'un roman est écrit à la première personne, le pronom de la première personne et la conjugaison du verbe ne suffisent pas à identifier le discours direct. Le corpus que nous utilisons ensuite contient un tel roman écrit à la première personne, *« C'est fou ce qu'on voit de choses dans la vie ! _DEBURON »*.

Le travail de Pouliquen et al. (2007) porte sur l'utilisation d'expressions régulières pour identifier des discours directs dans un contexte multilingue. Les trois règles générales qu'ils proposent dans l'article pourront être appliquées dans notre travail, avec des adaptations.

1. *quote-mark QUOTE quote-mark [,] verb [modifier] [determiner] [title] name*

Ex: *"blah blah", said again the journalist John Smith.*

2. name [, up to 60 characters,] verb [:|that] quote-mark QUOTE quote-mark

Ex: *John Smith, supporting AFG, said: "blah blah".*

3. quote-mark QUOTE quote-mark [; or ,] [title] name [modifier] verb

Ex: *"blah blah", Mr. John Smith said.*

Bien que ces chercheurs travaillent sur la presse, les trois règles qu'ils proposent ci-dessus sont toujours utiles pour l'identification automatique de discours direct dans notre tâche. Nous constatons que dans notre domaine de travail, le roman français, on peut essentiellement affirmer que l'apparition du discours direct est guidée par la ponctuation, qui correspond au *« quote-mark »* dans la règle proposée par Pouliquen et al. (2007), et nous pouvons modifier *« quote-mark »* pour répondre aux exigences de notre travail. Dans notre travail, nous n'avons pas besoin d'une telle décomposition pour les informations proposées par Pouliquen et al. (2007), telles que « *title* », « *name* », etc., mais nous considérons plutôt, par exemple, que « *Mr. John Smith said* » est un tout, et que tous les tokens de cet ensemble ne font pas partie du discours direct. Pouliquen et al. (2007) soulignent également l'importance du verbe, car il peut nous aider efficacement à trouver les parties hors du discours direct, donc les *verba dicendi* se révèleront très important pour notre travail

d'identification automatique du discours direct. Avec les trois règles proposées par Pouliquen et al. (2007), il est également évident que la position dans laquelle la partie hors discours direct apparaît est flexible. Dans une phrase, elle peut apparaître soit avant la partie du discours direct, soit après la partie du discours direct. C'est donc aussi une indication utile pour notre travail sur la façon dont nous devons identifier les parties qui appartiennent au discours direct, en tenant compte de la flexibilité de l'endroit où les parties hors discours direct apparaissent.

2. APPROCHES NEURONALES

Avec le développement rapide de l'apprentissage automatique au cours des dernières années, notamment dans le domaine de l'apprentissage profond, ces approches ont été de plus en plus utilisées en TAL. Avec l'apprentissage automatique, l'ordinateur extrait lui-même les caractéristiques souhaitées pour la tâche à partir des données prédéfinies par le développeur, il apprend ces caractéristiques de manière continue afin de découvrir des régularités dans ces caractéristiques qui répondent aux exigences de la tâche, pour par la suite faire des prédictions avec les nouvelles données. Dans le cas de l'apprentissage supervisé, le système doit apprendre un modèle qui lui permet d'inférer des informations à partir de données annotées par des humains. Contrairement à ce type d'apprentissage, dans le cas de l'apprentissage non supervisé, le système doit apprendre un modèle à partir de données non annotées. Une fois que le modèle a été entraîné, ses performances peuvent être testées en utilisant des données ne faisant pas partie de ses données d'entrainement. Si les performances ne sont pas satisfaisantes, le modèle peut être réentraîné et les paramètres ajustés pour obtenir un meilleur modèle.

Il y a peu, parmi les principales architectures utilisées pour l'apprentissage automatique, on pouvait mentionner les CNN (Réseau de neurones à convolution) et les RNN (Réseau de neurones récurrents). Les CNN basés sur la convolution présentent un avantage pour la reconnaissance de la structure de la tâche cible, tandis que les RNN présentent un avantage pour la modélisation de la reconnaissance des séquences, grâce à sa fonction de mémorisation du contexte. Les CNN et les RNN ont également leurs propres avantages lorsqu'ils sont appliqués à des tâches de TAL. L'article de Yin et al. (2017) présente les résultats des modèles CNN et RNN (GRU, LSTM) pour une série de tâches comprenant l'analyse des sentiments, la classification des phrases, la reconnaissance des entités, les systèmes de questions-réponses, etc. En général, les résultats des trois modèles

(CNN, GRU et LSTM) sont proches les uns des autres pour les tâches d'analyse des sentiments, les CNN ayant un avantage dans la classification des phrases, et les RNN ayant un avantage dans les tâches séquentielles comme la reconnaissance des entités nommées et les questions-réponses contextualisées.

Pour les tâches de reconnaissance d'entités nommées, les RNN étaient généralement préférés aux CNN jusqu'à l'arrivée des modèles Transformers. Les Transformers sont une classe récente de réseaux neuronaux auto-attentifs basés sur des séquences qui se sont avérés bien adaptés au texte et qui sont actuellement à l'origine d'avancées importantes dans TAL. Les Transformers sont des modèles qui utilisent des mécanismes d'attention pour augmenter la vitesse d'apprentissage du modèle, et ils peuvent être décrits comme des modèles d'apprentissage profond entièrement basé sur des mécanismes auto-attentifs. En raison de leur applicabilité au calcul parallélisé et de la complexité de leurs propres modèles, ils sont plus précis et plus performants que les RNN précédemment populaires. Un autre avantage est la possibilité de préentrainer de très vastes modèles, tels que BERT (Devlin et al. 2019) ou GPT[44] (Radford et al. 2019), sur une grande quantité de texte de façon totalement non supervisée, ce qui permet aux modèles d'extraire une représentation contextualisée des mots au sein d'une phrase (ce qu'on appelle des *words embeddings*) qui tient compte de leur environnement lexico-syntaxique. Typiquement, ce sont ces *embeddings* génériques et non supervisés qui sont utilisés pour réaliser des tâches spécifiques et supervisée, par *fine tuning*, telle que celle qui nous intéresse ici.

Nous présentons ici en termes généraux quelques travaux utilisant l'approche neuronale pour l'identification automatique des discours direct.

Le travail de Schöch et al. (2016) détermine si une phrase contient un discours direct ou non. La tâche est donc formulée comme une simple tâche de classification binaire. Leur corpus contient 127 romans français publiés entre 1840 et 1889. On peut distinguer trois sous-ensembles génériques, chacun étant représenté par une quarantaine de textes : la fiction romanesque générale s'oppose à des sous-genres spécifiques, le roman policier et le roman fantastique. Après avoir prétraité le texte, ils ont modélisé les caractéristiques utiles à la tâche de classification au niveau de la phrase. On peut distinguer les catégories suivantes : ponctuation, lexique, sémantique, morphologie et syntaxe. Ils ont utilisé un cadre

[44] GPT (Generative Pre-Training), un modèle proposé par OpenAI en 2018, utilise le modèle Transformer pour résoudre différents problèmes de langage naturel.

d'annotation et de classification développé par Markus Krug (Würzburg) enveloppant *LibSVM Support-Vector-Machine* (Chang & Lin 2011), *Maximum Entropy* (Nigam et al. 1999) et *Naïve Bayes* (John & Langley 1995) et implémenté dans *MALLET* (McCallum 2002). Les résultats expérimentaux ont montré que Random-*Forest* obtient les meilleurs résultats, même en excluant le signe vocal des caractéristiques (le tiret long), ils obtenent un score F1 de 0,924. Et ils concluent qu'il apparaît clairement que seuls quelques caractéristiques sont des indices forts du discours direct, à savoir (et sans surprise) le tiret cadratin à l'initiale. La plupart des autres caractéristiques, prises séparément, constituent des signaux faibles, mais deviennent pertinentes lorsqu'elles sont combinées. Ils concluent leur étude avec un bon résultat, un score F1 de 0.939.

Dans le travail de Jannidis et al. (2018), tout d'abord, les auteurs soulignent que l'annotation manuelle de grandes quantités de texte prend beaucoup de temps et est coûteuse. De plus, les annotations pour un type de texte peuvent ne pas être transférables à d'autres types, ce qui rend nécessaire l'utilisation de données annotées pour de nouveaux corpus. Les auteurs présentent donc une démarche faiblement supervisée. Leurs expériences sont basées sur trois corpus allemands. Le premier est un grand corpus contenant plus de 4600 romans du domaine public, y compris des textes de *TextGrid digital library*[45] et du *Projet Gutenberg*[46], nommé comme *le Corpus Public Domain, PD*. Le second contient plus de 800 textes de genres populaires actuels comme la romance, le crime ou la science-fiction (*Corpus Low Brow, LB*). Enfin, ils utilisent un corpus comprenant 200 romans nominés pour *German Book Prize* ou the *Georg Büchner Prize* (*Corpus High Brow, HB*). Ils présentent ainsi une méthode qui permet d'exploiter de grandes quantités de données grossièrement étiquetées de façon automatique, extraites de textes bruts. Lors de l'entraînement du classifieur, ils utilisent une règle simple basée sur les guillemets (parce qu'ils étudient la fiction allemande, où l'utilisation des symboles diffère de celle de la France) pour extraire des étiquettes « faibles » (les étiquettes sont de qualité incertaine, puisque non vérifiées à la main） en supposant que tout ce qui est écrit entre guillemets est du discours direct. Cependant, lors de l'entraînement du modèle, les auteurs choisissent de supprimer tous les guillemets du texte, afin d'éviter que les modèles entrainés ne se basent que sur les indices formels de qualification du discours direct (la ponctuation – les guillemets), et pour qu'ils prennent également en compte des

[45] https://textgrid.de/digitale-bibliothek
[46] http://gutenberg.spiegel.de/

signes implicites, comme l'utilisation de verbes à la première personne ou de mots de parole. Ils utilisent des réseaux neuronaux récurrents, qui sont capables de traiter des contextes relativement larges, car les marqueurs du discours direct peuvent être trouvés au début ou à la fin du discours direct. Enfin, dans la tâche de classification des phrases, ils obtiennent une précision de 0,85 en utilisant un classifieur à régression logistique ; dans la classification au niveau des mots, l'entraînement sur la moitié du *Kerncorpus*[47] et l'évaluation sur l'autre moitié, ils obtiennent une précision de 0,83. Lors de l'entraînement sur le *Kerncorpus* complet et de l'évaluation sur l'ALB[48] annoté manuellement, la précision atteint même 0,90.

Byszuk et al. (2020) ont choisi d'utiliser BERT (basé sur une architecture de transformer qui apprend les relations contextuelles entre les mots du texte) pour leur travail sur la reconnaissance directe des discours. Ils ont utilisé *European Literary Text Collection (ELTeC)*[49] comme corpus, qui devrait être composé d'environ 2 500 romans dans au moins 10 langues différentes. Leur travail peut être résumé par les trois étapes suivantes :

1) Extraction d'échantillons de la base de données et annotation manuelle de la section discours direct. Les annotations suivent les recommandations TEI, en marquant la section discours direct entre la balise *<said></said>*.

Ex :

<said> — ***Je suis du même avis*** *</said>, s'écria Benno Tönnchen avec empressement.*

2) Sur la base de l'étape 1, les données sont converties en un format d'étiquette et de label accepté par BERT. *« I »* pour le discours direct (direct speech) et *« O »* pour le discours indirect (indirect speech), avec *</p>* marquant la fin du paragraphe.

Ex :

<said> — ***Je suis du même avis*** *</said>, s'écria Benno Tönnchen avec empressement.*

---

[47] Un sous-ensemble du *corpus PD*, *Kerncorpus* est composé de 250 textes de haut et moyen niveau (ceux de *TextGrid digital library*), qui ont été édités manuellemnt.

[48] Un sous-ensemble plus petit du *Corpus LB*. Sélectionné 50 extraits de *low brow literature*. ALB est relativement biaisé en faveur des textes qui ne sont pas en discours direct, avec seulement environ 18% des tokens en discours direct.

[49] https://www.distant-reading.net/eltec/

⇒ *<said> — **Je suis du même avis** </said>, s'écria Benno Tönnchen avec empressement.</p>*

⇒ Je	I	s'écria	O
suis	I	Benno	O
du	I	Tönnchen	O
même	I	avec	O
avis	I	empressement	O
,	O	.	O

3) Les données de l'échantillon sont divisées en Train (80%), Test (10%) et Dev (10%) pour l'entraînement du modèle.

Le résultat final qu'ils ont obtenu est le score F1 final de leur modèle est de 0,873.

Toujours en utilisant BERT, Kurfali et al. (2020) adoptent une approche différente de celle de Byszuk et al. (2020). Conformément à l'idée de Jannidis et al. (2018), Kurfali et al. (2020) ont également choisi de supprimer la ponctuation dans l'espoir que le modèle apprenne les caractéristiques linguistiques du discours direct plutôt que la typographie. Ils ont utilisé le format BIO, en étiquetant chaque token comme : « *narration* », « *beginning of speech* » et « *inside speech* ». Ils ont utilisé un total de quatre corpus, RiQuA, SLäNDa, Redewiedergabe et QUAC. Les corpus ont ensuite été utilisés pour développer un classifieur au niveau du token pour identifier les discours directs sans ponctuations présentes. Puisque le travail de Kurfali et al. (2020) porte sur l'identification du discours direct multilingue, pour la reconnaissance de la ponctuation, ils font ce qui suit :

- Ils ne supposent pas à l'avance que les textes de différentes langues utilisent systématiquement un symbole particulier pour le discours direct. Cela est dû au fait que la représentation du discours direct diffère dans les différents codes linguistiques. Par exemple, un tiret peut être utilisé pour indiquer une ligne de conversation ou pour indiquer une pause nette (en anglais).

Ex: "I know you are still there – somewhere in the sky."

- Ils dressent une liste des ponctuations utilisées pour représenter le discours direct dans différents codes linguistiques («», *“ ”*, "" , – , —), afin de filtrer les textes collectés.

- Ils calculent d'abord la fréquence de chaque ponctuation possible dans la liste et filtrent le livre si l'un des ponctuations n'est pas dominant, c'est-à-dire s'il ne représente pas (> 90 %) de toutes les occurrences. La raison d'être de cette étape est d'éliminer les cas où plusieurs marqueurs peuvent être utilisés de manière interchangeable, et donc probablement pas de manière cohérente, ce qui conduirait à des annotations de mauvaise qualité.
- Ils filtrent ensuite les livres en se basant sur le ratio du nombre de tokens de parole par rapport à l'ensemble du livre. Plus précisément, ils comptent les tokens qui se trouvent entre les ponctuations trouvées dans la première étape et n'acceptent un livre que si ces tokens de parole constituent plus de 30% et moins de 50% du livre entier, afin d'obtenir un corpus équilibré.

Ceci est instructif pour notre travail, car le corpus utilisé dans le stage concerne différents romans et il peut y avoir des variations dans l'utilisation de la ponctuation pour le discours direct.

Contrairement à d'autres, Kurfali et al. (2020) sont conscients que le discours direct et ses marqueurs associés peuvent s'étendre sur plusieurs phrases. Par exemple : *... he, then, exclaimed…*. Par conséquent, ils divisent le texte en morceaux (*chunks*) de phrases. Il y a au maximum 100 tokens dans chaque *chunk*, et il n'y a aucun cas où la phrase est coupée en deux. Les phrases ne sont divisées en sous-phrases que lorsqu'elles dépassent 100 tokens. Les résultats finaux de leur expérience sont les suivants : le score moyen de F1 est 0.74 (sans le corpus QUAC) / 0.6375(avec QUAC).

3. SYNTHESE DES RESULTATS ET ANALYSE DES ERREURS

Bien qu'en général les résultats de l'identification pour le discours direct soient bons (voir le tableau ci-dessous), il existe encore des facteurs d'erreur d'identification qui méritent notre attention.

Travaux	Résultat	Genres de texte	Langue
Poulard et al. (2008)	Precision:89% Rappel: 93% F-Mesure:91%	Journaux francophones	Français
Schöch et al. (2016)	Le score F1 de 0.939	Roman	Français
Magdalena Barnas (2017)	Plus ou moins performantes dans 80 %	Roman	Français

Jannidis et al. (2018)	Précision au niveau des phrases :85% ; Précision au niveau des tokens: 83%(a moitié du Kerncorpus); 90%(le Kerncorpus complet et l'évaluation sur l'ALB annotée manuellement)	Roman	Allemand
Sini et al. (2018)	F-mesure :89.09	Corpus d'audiobooks	Français
Tu et al. (2019)	Gutenberg : 85.4% Précision au niveau des phrases ; 86.8% Précision au niveau des tokens DROC_red :80.5% Précision au niveau des phrases ; 80.4% Précision au niveau des tokens RW_fict :81.9% Précision au niveau des phrases ; 81.7% Précision au niveau des tokens RW_nonfict :60.8% Précision au niveau des phrases ; 53.4% Précision au niveau des tokens	Roman(Gutenberg、DROC_red2) RW (textes fictionnels+non-fiction texte)	Allemand
Kurfali et al. (2020)	Le score moyen de F1 : 0.74 (sans le corpus QUAC) / 0.6375(avec QUAC)	Roman(SLäNDa、RiQuA) Ressource pour la représentation de la parole, de la pensée et de l'écriture (Redewiedergabe) Texte de non-fiction (QUAC)	Multilingue (Redewiedergabe-Allemand QUAC-Portugais SLäNDa-suédois RiQuA-Anglais
Byszuk et al. (2020)	Le score F1 de 0.873 Rappel :0.874 Precision: 0.873	Roman	Multilingue (Anglais Français、Allemand、Italien 、Norvégien、Portugais、Roumain、Serbe 、Slovène)

Tableau 5. Synthèse des résultats pour l'identification du discours direct

Dans leur article, Schöch et al. (2016) soulignent qu'une des sources d'erreur peut être due à une segmentation imparfaite des phrases. Plusieurs caractéristiques utilisées pour définir et identifier le discours direct (points d'interrogation/exclamation, interjections, temps de verbe) peuvent également conduire à une attribution incorrecte, notamment dans le contexte de la narration homodiégétique, où le narrateur est impliqué dans l'intrigue de sorte que son discours de narrateur est similaire au discours direct. Enfin, les lettres (au sens

épistolaire) sont parfois confondues avec le discours direct, ce qui est logique étant donné que dans la plupart d'entre elles, une personne s'adresse à une ou plusieurs autres personnes.

Sini et al. (2018) concluent que les erreurs proviennent essentiellement de deux aspects :

1. Le verbe utilise la forme participiale au lieu de la conjugaison normale. Cela engendre une mauvaise localisation de l'incise qui identifie le discours direct.

Ex:

— cinq cents, vers à toute la classe! ***exclamé*** *d'une voix furieuse,* ***arrêt****a, comme le Quos ego, une bourrasque nouvelle.*

Une identification incorrecte situe l'incise au niveau de *« arrêta »* plutôt qu'avec la forme participiale *« exclamé »*.

2. Il est également possible que le logiciel utilisé soit incorrect dans son analyse syntaxique du texte. Bien qu'il s'agisse d'un problème lié au logiciel (lié au prétraitement), cela entraîne également des erreurs dans l'identification du discours direct à un stade ultérieur.

Byszuk et al. (2020) indiquent qu'en plus des journaux intimes et des lettres mentionnés par Schöch et al. (2016), les passages narratifs à la première personne peuvent également être confondus avec des discours directs. Jusqu'à présent, cette erreur est linguistiquement insoluble. Et avec les méthodes d'apprentissage automatique, l'apprentissage des modèles génère également du bruit, qui affecte l'efficacité de l'identification du modèle.

4. CONCLUSION

De façon générale, les deux approches présentent des avantages et des inconvénients. Les avantages des approches symboliques basées sur des règles sont qu'elles sont faciles à interpréter et qu'elles ne nécessitent pas d'élaborer des données d'apprentissage comme les approches neuronales. Cependant, les inconvénients sont que les coûts de développement sont généralement élevés et que l'annotation manuelle prend du temps et coûte cher. Si une approche neuronale est utilisée, il est possible de former un modèle adapté à la reconnaissance multilingue des discours directs. Grâce à l'apprentissage automatique, le réseau neuronal peut trouver lui-même les caractéristiques pertinentes pour le discours

direct. Cependant, cette approche n'est pas aussi facile à utiliser que l'approche symbolique, qui nécessite que l'utilisateur ait des connaissances en apprentissage automatique, en réseaux neuronaux, etc. Et l'entraînement et le réglage du modèle prennent également beaucoup de temps.

Partie 3
-
EXPERIMENTATIONS

Chapitre 5. RESSOURCES ET DONNEES

Pour notre travail, nous disposons de deux ressources initiales : un corpus de romans de sous-genres diversifiés et un corpus de 15 romans intégrant une annotation du discours direct, qui a été élaboré au LIDILEM par M. Barnas, que nous désignerons dans la suite par le terme « corpus initial ». Une approche que nous pouvons donc envisager consiste à concentrer notre recherche sur les parties des romans du corpus initial qui sont marquées comme discours direct, afin d'extraire les caractéristiques qui peuvent être utilisées pour l'identification automatique du discours direct dans le corpus romanesque. Dans cette optique, il nous faut tout d'abord bien comprendre les données que ce corpus initial fournit pour notre travail.

Dans ce chapitre, nous décrivons premièrement les deux corpus que nous avons utilisés, puis nous explorons en détail les données fournies par le corpus initial.

1. DESCRIPTION DU CORPUS

1.1. Corpus initial

1.1.1. Description et composition

Le corpus initial dont nous disposons comporte 15 romans, pour un total de 38 843 paragraphes et de 275 368 phrases.

Nom du roman	Auteur	Date de publication	Genres[50]
C'est fou ce qu'on voit de choses dans la vie	Nicole de Buron	2006	SENT
Échine	Philippe Djian	1988	GEN
Etoiles en perdition	Pierre Barbet	1970	SF
Fleur de pavé	Pascal Lainé	1996	HIST
Flibustière	Johan Heliot	2012	HIST
Involution	Johan Heliot	2014	SF
L'enfer	René Belletto	2000	POL
L'oeuvre au noir	Marguerite Yourcenar	1968	GEN
La 7e femme	Frédérique Molay	2006	POL

[50] *GEN* : littérature générale, *SENT* : romans sentimentaux, *POL* : policiers, *HIST* : historiques, *SF* : science-fiction

Le zèbre	Alexandre Jardin	1988	SENT
Dans les bois éternels	Fred Vargas	2006	POL
Quelqu'un marchait sur ma tombe	Frédéric Dard	2016	POL
Le dragon aux plumes de sang	Pierre Bordage	2003	SF
Le premier jour	Marc Levy	2009	SENT
La première nuit	Marc Levy	2009	SENT

Tableau 2. La description du corpus initial

1.1.2. Évaluation la Distribution du discours direct

Dans chaque roman, les phrases ou passages qui contiennent du discours direct sont marqués comme *'type=dd'* par M. Barnas (2017). Afin de donner de la diversité à la recherche du discours direct, M. Barnas (2017) a choisi une variété de genres de romans, et différents auteurs, pour composer son corpus [51]. Afin d'examiner dans quelle mesure les caractéristiques du discours direct contenues dans les romans sont utilisées pour que les caractéristiques extraites soient génériques. Nous devons évaluer la couverture du discours direct de chaque roman, qui peut être utilisée pour identifier automatiquement le discours direct dans d'autres romans.

Pour ce faire, nous avons calculé le nombre de phrases de discours direct contenues dans chaque roman. Les résultats sont présentés dans le tableau ci-dessous.

Nom du roman	Nombre de phrases du discours direct	Nombre de phrases total	Pourcentage (par phrases)
C'est fou ce qu'on voit de choses dans la vie	1408	5202	27.07%
Échine	2572	8489	30.30%
Etoiles en perdition	1986	3340	59.46%
Fleur de pavé	3537	8924	39.63%
Flibustière	1765	4414	39.99%
Involution	1426	3062	46.57%

[51] Voir la Table 2 pour des informations détaillées

L'enfer	3215	8488	37.88%
L'oeuvre au noir	1324	4782	27.69%
La 7e femme	3462	6832	50.67%
Le zèbre	450	2833	15.88%
Bois éternels	7476	62623	11.94%
Quelqu'un marchait sur ma tombe	1693	3399	49.81%
Le dragon aux plumes de sang	2992	39756	7.53%
Le premier jour	4260	36899	11.55%
La première nuit	2733	76325	3.58%
TOTAL	**40299**	**275368**	**14.63%**

Tableau 3. Le nombre de phrases de discours direct (par phrase) du corpus initial

Les résultats de cette statistique montrent que le degré d'inclusion du discours direct dans les différents romans varie. Certains romans contiennent plus de discours direct, comme « *Etoiles en perdition* », qui atteint 59,46%, tandis que d'autres en contiennent moins, comme La première nuit, qui n'a que 3,58%. Par conséquent, nous ne pouvons pas simplement choisir un roman au hasard pour extraire les caractéristiques de discours direct, ni pour évaluer résultat de notre travail. Ce n'est pas la seule raison. Il peut également y avoir des différences dans la structure du discours direct, et nous espérons que les romans sélectionnés puissent contenir une variété de cas de figure.

1.2. Corpus Phraseorom

Nous avons utilisé le corpus *Phraseorom*[52], initialement constitué pour le projet de *PhraseoRom* (2016-2020). « *Le principal objectif du projet PhraseoRom a été d'élaborer une typologie structurelle et fonctionnelle des constructions lexico-syntaxiques (CLS) spécifiques au discours romanesque francophone, anglophone et germanophone à partir des années 1950* » (Diwersy et al., 2021).

1.2.1. Composition

Le corpus comporte environ 395 millions de mots et se divise en trois corpus linguistiques : anglais, français et allemand.

[52] Ce corpus est principalement utilisé dans la section d'apprentissage automatique

Les œuvres constitutives du corpus français ont été réparties en 6 sous-genres : littérature générale (ou « blanche »), romans sentimentaux, policiers, historiques, de science-fiction, fantasy (Diwersy et al., 2021).

Le but de notre projet est d'annoter le discours direct dans ce corpus, à partir du meilleur modèle que nous aurons élaboré à l'issue de notre travail.

2. DESCRIPTION DES DONNÉES

Comme mentionné ci-dessus, nous utilisons le corpus initial comme objet de recherche dans notre travail. Nous prenons donc les données du corpus initial comme données de base pour le développement et l'évaluation de notre premier système.

2.1. Format

Tout d'abord, chaque roman du corpus initial est stocké au format XML. Les romans sont divisés par paragraphe, en utilisant un ensemble de balises <*p*></*p*> pour représenter un paragraphe. Chaque paragraphe contient une ou plusieurs phrases, chacune d'entre elles étant représentée par un ensemble de balises <*s*></*s*>. Il existe donc des cas où un ensemble de balises <*p*></*p*> contient une ou plusieurs ensembles de balises <*s*></*s*>.

Les mots contenus dans un ensemble de balises <*s*></*s*> constituent le contenu de cette phrase. Ainsi, l'ensemble du contenu d'un roman est constitué de tous les mots contenus dans toutes les balises <*s*></*s*> de ce roman. Pour notre travail, l'identification automatique du discours direct, il s'agit de lire tous les mots contenus dans les balises <*s*></*s*> et de déterminer s'ils appartiennent au discours direct.

```
1  <p type="" id="p0" class="copyr"> <s type="" id="s1">ISBN : 978-2-266-16638-6\n    </s></p>
2  <p type="" id="p1" class="copyr"> <s type="" id="s2">Plon, 2006.\n    </s></p>
3  <p type="" id="p2" class="titre1b">    <s type="" id="s3">Souvenirs vrais et faux\n    </s></p>
4  <p type="" id="p3" class="titnom"> <s type="" id="s4">C'EST FOU CE QU'ON VOIT DE CHOSES DANS LA VIE !\n    </s></p>
5  <p type="" id="p4" class="titaut"> <s type="" id="s5">NICOLE DE BURON\n    </s></p>
6  <p type="" id="p5" class="dedi">   <s type="" id="s6">Je dédie ce livre à « Aliénor », marquise de P., à qui je dois tant.\n    </s></p>
7  <p type="" id="p6" id="calibre_link-24" class="calibre" dir="ltr"> <s type="" id="s7">La vie chez mes grands-parents\n    </s></p>
8  <p type="" id="p7" dir="ltr" class="calibre" id="calibre_link-25"> <s type="" id="s8">1 Départ en vacances\n    </s></p>
9  <p type="" id="p8" class="calibre5">   <s type="" id="s9">En route vers trois mois de bonheur, Madame Minerva !\n    </s></p>
10 <p type="" id="p9" class="calibre5">   <s type="" id="s10">Un roulement d'enfer réveilla tout le quartier, y compris le petit Régis D. </
11 <p type="" id="p10" class="calibre5">  <s type="" id="s15">Je feuilletais avec enchantement les gros livres roses dorés sur tranches de J
12 <p type="" id="p11" class="calibre5">  <s type="" id="s17">Joseph (en remontant dans sa chère Minerva) jeta un coup d'oeil pour vérifier
13 <p type="" id="p12" class="calibre5">  <s type="" id="s21">Il prit néanmoins le temps de vérifier que l'immense malle en cuir noir portan
14 <p type="dd" id="p13" class="calibre5">    <s type="" id="s24">— Bien, monsieur le baron, approuva joyeusement ce dernier.\n    </s></p>
15 <p type="dd" id="p14" class="calibre5">    <s type="" id="s25">— Allez, Joseph ! </s><s type="" id="s26">On y va...\n    </s></p>
```

Figure 5. Exemple du format XML du corpus initial

2.2. Introduction aux attributs utilisés

Il est facile de voir dans la figure ci-dessus que chaque ensemble de balises <*p*> et <*s*> possède des attributs spécifiques. La description de ces attributs nous permettra de mieux comprendre et d'approfondir les informations fournies par le corpus initial, ce qui sera

bénéfique pour notre travail dans les parties suivantes. Nous commencerons par les attributs que possèdent les balises <*p*> et <*s*>.

2.2.1. L'attribut type

Nous présentons tout d'abord l'attribut *type*. L'attribut type est utilisé pour indiquer si les balises <*p*> ou <*s*> contiennent du discours direct.

Pour la balise <*p*>, il n'y a que deux possibilités pour la valeur de l'attribut type comme suit.

- **Type = "dd"**

Il s'agit d'une représentation non ambiguë par rapport au *type=""* suivant, qui indique une signification unique et formelle. La valeur de l'attribut *type* est *"dd"*, ce qui signifie que toutes les phrases de ce paragraphe appartiennent au discours direct. C'est-à-dire que tous les balises <*s*> contenues dans cette balise <*p*> sont le discours direct.

Il convient de noter que lorsque l'attribut type d'une balise <p> est marqué comme *type="dd"*, le type de la balise <*s*> contenue dans la balise <p> n'est pas marqué comme *type="dd"*, ce qui est dû à la caractéristique de formatage XML. Pour le format XML, la balise <s> est comme un enfant de la balise <*p*>, et la balise enfant hérite des attributs de la balise de niveau supérieur. Dans ce cas, même si l'attribut type de la balise <*s*> n'est pas affecté de *type="dd"*, son contenu qu'elle contient appartient toujours au discours direct. Dans nos travaux ultérieurs, nous devons tenir compte de cette situation et pas simplement regarder l'attribut *type* de la balise <*s*> pour déterminer si la phrase appartient au discours direct. Sinon, cela entraînerait la perte d'une partie du discours direct.

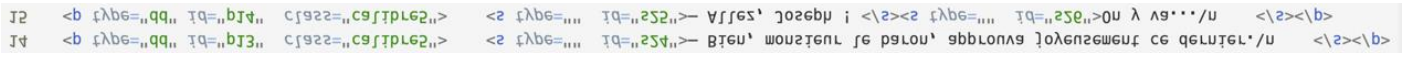


Figure 6. Exemple du *« type=dd » pour balise <p>*

- **Type=""**

Aucune valeur n'est attribuée à l'attribut *type*, ou nous pouvons l'exprimer en langage de programmation, c'est-à-dire que la valeur de l'attribut type est None. Dans le cas de *type=""*, cela signifie généralement que le contenu de cette balise <*p*> ne fait pas partie du discours direct.

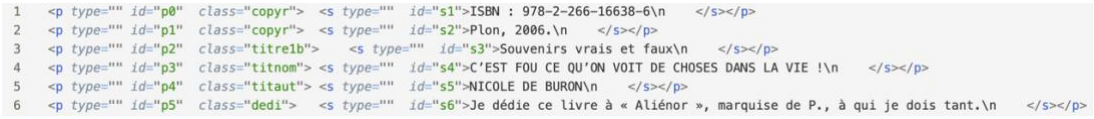


Figure 7. Exemple du *« type="" » pour balise <p>*

Comme nous l'avons expliqué plus haut, l'attribut *type* de la balise <*p*> ne sera marqué comme dd que si tout le contenu de la balise <*p*> fait partie de *dd*. En d'autres termes, s'il existe une balise <*s*> qui ne fait pas partie du discours direct parmi toutes les balises <*s*> contenues dans cette balise <*p*>, l'attribut *type* de cette balise <p>ne peut pas être marqué comme *type="dd"*. Ainsi, la situation suivante se produit : lorsque l'attribut *type* d'une balise<*p*> est comme *type=''*, les balises <*s*> qu'elle contient peuvent apparaître de deux façons : soit l'attribut *type* de la balise <s> est le même que l'attribut *type* de la balise <*p*>, comme *type=''*, soit l'attribut *type* de la balise <*s*> ne correspond pas à l'attribut *type* de la balise<*p*>, l'attribut *type* de la balise <*s*> étant *type="dd"*.

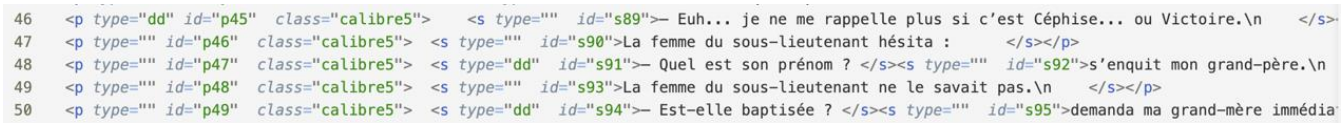


Figure 8. Exemple du *« type="" » pour balise <p>*

2.2.2. L'attribut id

Le second est l'attribut *id*. L'attribut *id* est utilisé pour enregistrer et représenter le nombre de paragraphes et de phrases dans chaque roman.

Pour la balise <*p*>, l'attribut id indique dans quel paragraphe se trouve ce roman. Ce que nous devons noter ici, c'est que la valeur de l'attribut id de la balise <*p*> est augmenté à partir de 0, donc le paragraphe réel doit être le numéro d'affichage de l'attribut id plus un. Par exemple, si *id="p4"* dans la balise <*p*>, il s'agit en fait du cinquième paragraphe du roman, et ainsi de suite.

Pour la balise <*s*>, l'attribut *id* indique de quelle phrase du roman il s'agit. Contrairement à l'attribut id porté par la balise <*p*>, la valeur de l'attribut id de la balise <*s*> est augmentée à partir de 1, de sorte que l'attribut id indique la situation réelle. Par exemple, si *id="s5"* dans la balise <*s*>, il s'agit de la cinquième phrase du roman, et ainsi de suite.

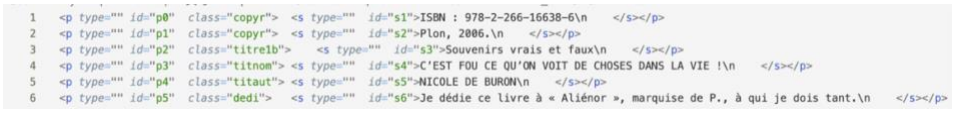


Figure 9. Exemple de l'attribut *id*

Chapitre 6. PRETRAITEMENT

Dans ce chapitre, notre objectif principal est de prétraiter les données du corpus initial fourni par M. Barnas (2017). Nous décrivons d'abord nos opérations de conversion de format du corpus initial. Ensuite, nous détaillons la manière de traiter nos données afin d'obtenir les données requises pour notre tâche de reconnaissance automatique du discours direct.

1. CONVERSION DE FORMAT DES DONNEES

Au chapitre 5, nous avons appréhendé de manière générale le format du corpus initial et la composition de son contenu. Il est facile de voir que certains de ces attributs n'ont aucun effet sur notre tâche de reconnaissance automatique du discours direct, comme l'attribut *class*, l'attribut *dir* et ainsi de suite. La présence ou l'absence de ces attributs ne fait aucune différence dans les résultats de la reconnaissance, et nous pouvons les appeler « *informations inutiles* ».

1.1. Supprimer les informations inutiles

Afin de rendre les informations fournies par nos données plus ciblées sur notre tâche d'identification automatique du discours direct, nous avons d'abord choisi de supprimer ces informations inutiles.

Pour l'opération de suppression ici, nous avons utilisé la fonction remplacer fournie dans le logiciel *Visual Studio Code*. Nous recherchons d'abord les attributs à supprimer, l'attribut *class*, l'attribut *dir* et l'attribut *id* qui n'est présent que lorsque l'attribut class est calibre. Il est important de noter que pour s'assurer que les résultats que nous recherchons correspondent à nos exigences, nous devons sélectionner l'option mot entier[53]. Après avoir trouvé ces attributs qui doivent être remplacés, nous utilisons la fonction remplacer pour remplacer ces attributs par *None*, ce qui permet la suppression de ces attributs. Le corpus de données initial après cette étape est présenté ci-dessous.

[53] Utilisez cette option pour vous assurer que votre recherche retourne uniquement des correspondances de mots entiers.
https://docs.microsoft.com/fr-fr/visualstudio/ide/find-in-files?view=vs-2022

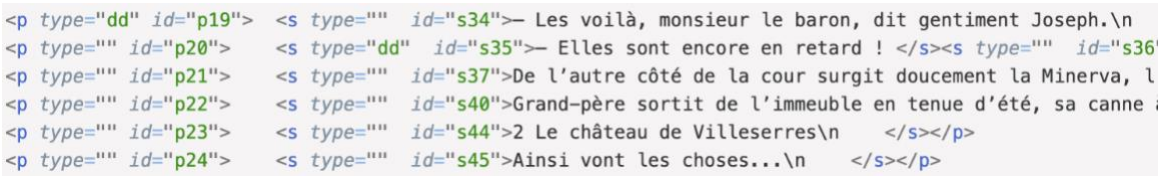


Figure 10. L'exemple du roman *« C'est fou ce qu'on voit de choses dans la vie »*

1.2. Convertir le format

Pour le corpus initial, nous savons que chaque balise <*p*> représente un paragraphe et qu'une balise <*s*> représente une phrase. Une balise <*p*> peut contenir une ou plusieurs balises<*s*>, et les balises <*s*> sont des sous-balises des balises <*p*>. Cependant, dans le corpus initial, chaque balise <*p*> et les balises <*s*> qu'il contient sont tous dans la même ligne, ce qui n'est pas pratique pour notre vue. Pour donner un sens plus intuitif aux données, nous voulions des sauts de ligne et des retraits entre les balises <*p*> et les balises <*s*>, en d'autres termes, une ligne distincte pour chaque balise <*s*>. Nous convertissons donc le corpus initial, après avoir éliminé les informations inutiles, son fonctionnement peut être résumé, dans les grandes lignes, à un procédé en trois étapes.

1.2.1. Un saut de ligne pour la balise <s>

Ici, nous utilisons toujours la fonction de remplacer fournie par le logiciel Visual Studio Code. La première étape consiste à trouver la balise <*s*>. Avant de pouvoir compléter le saut de ligne, nous devons séparer la balise <*s*> de la balise <*p*> sur des lignes distinctes, nous commençons donc par chercher le début de la balise <*s*>, *« <s »*. L'option mot entier mentionné en 1.1 est toujours sélectionné pour la requête. Comme nous voulons que chaque balise s soit sur une ligne séparée, nous devons ajouter un caractère de nouvelle ligne à tous les « <*s* » que nous trouvons. Nous utilisons la fonction remplacer pour transformer le « <*s* » en « \n<s »[54]. Pour que « *n* » représente un caractère d'échappement, plutôt qu'une simple chaîne de caractères, nous devons également sélectionner l'option expression régulière en plus de l'option mot entier. Le résultat de cette étape est présenté ci-dessous.

[54] « \n », Caractère d'échappement, représente la nouvelle ligne

```
<p type="" id="p0">
<s type=""  id="s1">ISBN : 978-2-266-16638-6\n    </s></p>
<p type="" id="p1">
<s type=""  id="s2">Plon, 2006.\n    </s></p>
<p type="" id="p2">
<s type=""  id="s3">Souvenirs vrais et faux\n    </s></p>
<p type="" id="p3">
<s type=""  id="s4">C'EST FOU CE QU'ON VOIT DE CHOSES DANS L
<p type="" id="p4" >
<s type=""  id="s5">NICOLE DE BURON\n    </s></p>
```

Figure 11. L'exemple du roman *« C'est fou ce qu'on voit de choses dans la vie »*

1.2.2. Le retour pour la balise <s>

Après avoir terminé l'étape 1, nous pouvons effectuer un retour sur toutes les balises <*s*>. L'opération ici est la même que celle du saut de ligne de la balise <*s*>, la seule différence est que le caractère d'échappement « *n* » est remplacé par le caractère d'échappement « *t* »[55], nous ne les énumérons pas ici, pour plus d'informations spécifiques, se trouvent dans 1.2.1. L'entrée pour la fonction replacer reste la même (« <*s* »), « *n*<*s* » est remplacé par « *t*<*s* ». Le résultat est présenté ci-dessous.

```
<p type="" id="p0">
    <s type=""  id="s1">ISBN : 978-2-266-16638-6\n    </s></p>
<p type="" id="p1">
    <s type=""  id="s2">Plon, 2006.\n    </s></p>
<p type="" id="p2">
    <s type=""  id="s3">Souvenirs vrais et faux\n    </s></p>
```

Figure 12. L'exemple du roman *« C'est fou ce qu'on voit de choses dans la vie »*

*1.2.3. Manipulation de la balise *

Après les deux premières étapes, les balises <*s*> ont été traités comme on pouvait s'y attendre. La seule chose qui doit être traitée maintenant est la dernière balise <s> contenue dans chaque balise <*p*> (qui peut aussi être la première balise <*s*>, car certaines balises ne contiennent qu'une seule balise <*s*>. Dans ce cas, la balise <s> est à la fois la première et la dernière balise <*s*>). Nous constatons que la fin de la dernière balise </*s*>, est suivie d'une balise </*p*> indiquant la fin d'un paragraphe. Il suffit donc de faire un saut de ligne sur cette balise </*p*>, en gardant le même format que la balise <*p*>.

---

[55] Correspond à une tabulation
https://docs.microsoft.com/fr-FR/dotnet/standard/base-types/regular-expression-language-quick-reference

La procédure exacte est la même que l'opération de saut de ligne pour la balise <*s*> en *1.2.1*. L'entrée pour la fonction replacer est * *, « *\n<s* » est remplacé par « *\n * »[56].

```
<p type="" id="p0">
    <s type=""  id="s1">ISBN : 978-2-266-16638-6\n    </s>
</p>
<p type="" id="p1">
    <s type=""  id="s2">Plon, 2006.\n    </s>
</p>
<p type="" id="p2">
    <s type=""  id="s3">Souvenirs vrais et faux\n    </s>
</p>
```

Figure 13. L'exemple du roman *« C'est fou ce qu'on voit de choses dans la vie »*

1.3. Ajouter "dd"

Il reste encore un problème à résoudre, celui de l'attribut *type*. Comme mentionné au chapitre 5, il y a une ambiguïté lorsque l'attribut type de la balise <*s*> est « *type=""* », et que nous ne pouvons pas savoir s'il appartient au discours direct, et devons utiliser l'attribut type de la balise <*p*> pour le vérifier. Pour éliminer cette incertitude, nous avons l'intention de marquer "type=dd" à l'attribut type de toutes les balises <*s*> contenues lorsque l'attribut *type* de la balise <*p*> est "type=dd". De cette façon, La signification exprimée par l'attribut type de la balise <*s*> n'est pas ambiguë ; lorsque « *type="dd"* », cela signifie que la phrase représentée par cette balise <*s*> appartient au discours direct ; lorsque type="", cela signifie que la phrase représentée par cette balise <*s*> n'appartient pas au discours direct.

Le script consistant à projeter l'attribut "dd" de l'élément vers l'élément <s>, dont le pseudo-algorithme est présenté en Annexe 1, a été réalisé en Python avec l'aide du module *xml.dom.minidom* pour parser les fichiers du corpus. Il prend en entrée des fichiers XML du corpus et donne en sortie des nouveaux fichiers XML. L'attribut de type de toutes les balises <*s*> du nouveau fichier qui répondent aux exigences se voit attribuer la valeur « *type="dd"* ». Son fonctionnement peut être résumé, dans les grandes lignes, à un procédé en trois étapes.

Avant de pouvoir commencer à exécuter le script, nous devons également travailler sur les fichiers du corpus. Comme le fichier de notre base de données n'est pas vraiment un fichier au format XML, il lui manque un élément racine. Les balises <*p*> et <*s*> que nous avons mentionnées précédemment sont des balises enfants. La balise p n'est pas vraiment l'élément racine. Un fichier au format XML dépourvu d'un élément racine signalera une

[56] Sélectionner l'option expression régulière en plus de l'option mot entier

erreur à l'étape de lecture du script lors de son exécution ultérieure. Pour éliminer ce problème, nous ajoutons simplement un ensemble de balises d'élément racine <*div*> et </*div*> au début et à la fin du document.

1.3.1. Étape de lire un fichier

Les fonctions du module *xml.dom.minidom* sont utilisées pour ouvrir le fichier et effectuer l'opération de lecture. Conformément au format du fichier, nous parcourons les balises <*p*> afin de lire le texte complet.

1.3.2. Étapes d'étiquetage

La deuxième étape consiste à déterminer la balise <*p*> et à étiqueter l'attribut type à la balise <*s*>. Lorsque nous parcourons chaque balise<*p*> à l'étape précédente, nous devons déterminer l'attribut de type de la balise <*p*> et utiliser l'attribut de type de la balise <*p*> pour décider d'attribuer une valeur à l'attribut de type de la balise <*s*> qu'elle contient.

Par exemple, en supposant que la valeur de l'attribut type de la balise <*p*> que nous lisons actuellement est « *dd* », nous devons affecter « *dd* » à l'attribut de type de toutes les balises <*s*> qu'elle contient ; inversement, si l'attribut de type de la balise<*p*> n'est pas *« type = dd »*, nous n'effectuons aucune opération sur les balises <*s*> qu'elle contient, jusqu'à la fin du fichier.

1.3.3. Étape de création de la sortie

Enfin, la dernière étape consiste en la création de la sortie en fonction des résultats de l'étape de marquage précédente. La sortie prend la forme d'un fichier *.xml* contenant l'attribut type de toutes les balises <*s*> correspondants est marqué comme *« type=dd »*.

```
<p type="dd" id="p13">
    <s type=""  id="s24">— Bien, monsieur le baron, approuva joyeusement ce dernier.\n    </s>
</p>
<p type="dd" id="p14">
    <s type="dd"  id="s25">— Allez, Joseph ! </s>
    <s type="dd"  id="s26">On y va...\n    </s>
</p>
<p type="" id="p15">
    <s type=""  id="s27">Grand-père décrocha alors un cornet en corne couleur crème, orné de volu
</p>
```

Figure 14. L'exemple du roman *« C'est fou ce qu'on voit de choses dans la vie »*

Dans l'ensemble, après le processus de formatage, le nouveau fichier est toujours au format *xml*, mais la représentation globale est plus intuitive et plus claire et il n'y a aucune

ambiguïté sur le fait que la phrase soit une partie du discours direct ou non. Cela constitue une bonne base pour le traitement ultérieur des données.

2. ANNOTATION AUTOMATIQUE DES DONNEES

Les informations fournies par le corpus initial après formatage sont sans ambiguïté, de sorte que le corpus initial peut être traité ultérieurement pour répondre aux besoins de notre travail.

2.1. Tokenisation et annotation avec Stanza

Comme nous l'avons mentionné précédemment dans la Partie II, nous savons qu'une phrase peut contenir des parties du discours direct, des parties en incise, et des parties du discours indirect. Nous espérons que grâce à ce travail, nous pourrons identifier efficacement et automatiquement les parties de la phrase qui appartiennent au discours direct, c'est-à-dire les tokens de la phrase qui représentent du discours direct, tandis que les tokens restants, qu'ils représentent des parties incise ou du discours indirect, ne seront pas marqués comme discours direct. Pour pouvoir marquer différemment les différentes parties, il nous font donc commencer par tokeniser, c'est-à-dire segmenter les phrases en tokens.

Pour répondre aux exigences de notre travail, il est clair que les données basées sur les phrases du corpus initial fourni par Barnas (2017) ne sont pas adéquates. Nous ne pouvons pas traiter chaque token pour déterminer si ce token appartient à notre catégorie de reconnaissance automatique du discours direct. La première chose que nous devons faire est de diviser chaque phrase en tokens, en d'autres termes, de convertir les données basées sur les phrases du corpus initial fourni par Barnas (2017) en données basées sur les tokens.

Pour le travail de la tokenisation, nous utilisons une librairie *Python* dans le TAL, appelée *Stanza*, développée par le *Stanford NLP Group*. Nous commençons par une brève introduction à *Stanza*.

Stanza est une librairie *Python* précise et efficace pour l'analyse linguistique de nombreuses langues humaines. *Stanza* contient des outils permettant de convertir une chaîne de caractères contenant du texte en langue humaine en listes de phrases et de tokens, de générer les formes de base de ces mots (les lemmes), leurs parties du discours et leurs caractéristiques morphologiques, de donner une analyse syntaxique de dépendances de structure et de reconnaître les entités nommées. La librairie est conçue pour être universelle, et elle possède des modèles pour plus de 70 langues. Nous utiliserons les fonctions suivantes dans *Stanza* pour notre travail : *tokenization*, *lemmatization* et *part-of-speech (POS)*.

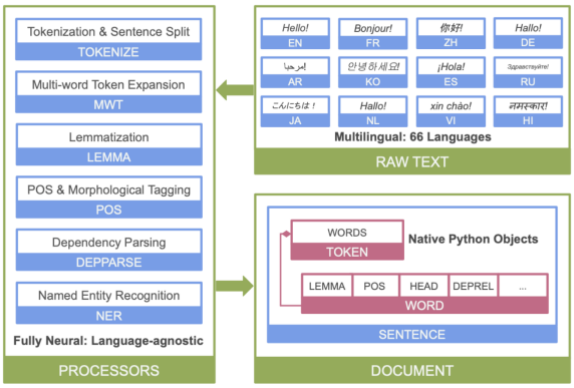


Figure 15. Une vue d'ensemble du pipeline NLP à réseaux neuronaux de *Stanza*[57]

Nous avons utilisé *Stanza* pour diviser les phrases du corpus initial en tokens. Conformément aux usages tel nous a marqué les tokens avec la balise <*t*>, chaque <*t*> représentant un token.

2.2. Ajouter l'attribut

Pour marquer les tokens, nous avons utilisé la balise <t>.

Pour chaque balise <*t*>, les attributs suivants ont été ajoutés afin d'enrichir l'annotation et de faciliter notre travail ultérieur sur la reconnaissance automatique du discours direct. Nous avons ainsi ajouté l'attribut *num*, l'attribut *l*, l'attribut *c* et l'attribut *f* à partir des annotations fournies par *Stanza*.

2.2.1. L'attribut id

De même que pour la balise <*p*> et la balise <*s*>, nous ajoutons un attribut id à chaque balise <*t*>, qui est utilisé pour enregistrer et représenter le nombre de tokens dans chaque roman. La valeur de l'attribut id de la balise <*t*> est augmenté à partir de 1, de sorte que l'attribut id indique la situation réelle. Par exemple, si « *id="t5"* » dans la balise <*t*>, il s'agit de la cinquième token du roman, et ainsi de suite.

2.2.2. L'attribut num

L'attribut *num* est également utilisé pour enregistrer et représenter le nombre de tokens. Cependant, contrairement à l'attribut id ci-dessus, la portée de l'attribut *num* est limitée à une seule phrase, c'est-à-dire une balise <*s*>. Il indique le numéro du token dans

[57] https://stanfordnlp.github.io/stanza/, 2020

une phrase, et peut servir d'identifiant pour les dépendances (bien qu'ici nous ne les ayons pas utilisées).

2.2.3. L'attribut l

La valeur de l'attribut *l* est obtenue à partir de la fonction *lemmatization* de *Stanza*. L'attribut *l* indique quel est le lemme du token représenté par cette balise <*t*>. Par exemple : « *l="le"* » dans la balise <*t*> signifie que le lemme de ce token est « *le* », mais ce token peut être « *la* » ou « *le* », et ainsi de suite.

2.2.4. L'attribut c

Comme l'attribut *l* mentionné ci-dessus, l'attribut *c* est obtenu par *Stanza* (la fonction pos). L'attribut c indique la catégorie du token représenté par cette balise <*t*> (son étiquette POS suivant les conventions UD, cf. https://universaldependencies.org/u/pos/). Par exemple : « *c="VERB"* » dans la balise <*t*> signifie que ce token est un verbe, et ainsi de suite.

2.2.5. L'attribut f

L'attribut f est obtenu par la fonction *feats* de *Stanza*, qui indique les informations sur les caractéristiques morphologiques du token représenté par la balise <*t*>. Par exemple : « *f="Gender=Masc|Number=Sing"* » dans la balise <*t*> signifie que ce token est masculin, nombre singulier, et ainsi de suite, avec les étiquettes *Universal features* de UD (https://universaldependencies.org/u/feat/index.html).

2.2.6. L'attribut type

L'attribut *type* exprime la même signification qu'avec les deux balises précédentes (<s> et <*p*>), l'attribut type est utilisé pour indiquer si le token appartient au discours direct ou non. La valeur de l'attribut type dans une balise <*t*> est liée à la balise <*s*> à laquelle elle correspond. En fonction de l'héritage, si l'attribut type de la balise <s> est « *type="dd"* », alors la valeur du type de toutes les sous-balises <*t*> qu'elle contient se voit attribuer la valeur « *dd* », de sorte que la valeur *type=""* ne soit pas ambiguë

```
<p id="p3">
    <s id="s4">
        <t id="t6" num="1" l="édition" c="NOUN" e="" f="Gender=Fem|Number=Plur" type="">Éditions</t>
        <t id="t7" num="2" l="il" c="PRON" e="" f="Number=Sing|Person=1|PronType=Prs" type="">J'</t>
        <t id="t8" num="3" l="avoir" c="AUX" e="" f="Mood=Ind|Number=Sing|Person=1|Tense=Pres|VerbForm=Fin" type="">ai</t>
        <t id="t9" num="4" l="lire" c="VERB" e="" f="Gender=Masc|Number=Sing|Tense=Past|VerbForm=Part" type="">lu</t>
        <t id="t10" num="5" l="," c="PUNCT" e="" f="" type="">,</t>
        <t id="t11" num="6" l="2014" c="NUM" e="" f="Number=Plur" type="">2014</t>
    </s>
</p>
```

Figure 16. L'exemple du roman « *C'est fou ce qu'on voit de choses dans la vie* »

2.3. Script

Le script du traitement ultérieur du corpus initial, dont le pseudo-algorithme est présenté en Annexe 2, a été réalisé en *Python* avec l'aide du module *xml.dom.minidom* pour parser les fichiers du corpus initial, en utilisant *Stanza* pour effectuer certains traitements des phrases tels que la *tokenisation* etc. Il prend en entrée des fichiers XML du corpus initial et donne en sortie des nouveaux fichiers XML, dans lequel chaque phrase du corpus initial (c'est-à-dire chaque balise <*s*>) est divisée en un token par la fonction de *tokenisation* de *Stanza*. Il est important de noter que, puisque *Stanza* est disponible dans de nombreuses langues, nous devons d'abord télécharger une librairie en français dans *Stanza* pour notre travail.

Pour faciliter le traitement ultérieur des données, une nouvelle balise, la balise <*t*>, est ajouté au fichier XML de sortie. Les balises <*t*> existent en tant que sous-balises des balises <*s*>, et chaque balise <*t*> contient un et un seul token. En d'autres termes, toutes les balises <*t*> contenues dans une balise <*s*> s'additionnent pour former le contenu de la phrase. Et pour chaque balise <*t*>, il y a des attributs *id, num, l, c* et *f*. Afin de simplifier le fichier, dans la nouvelle sortie XML, nous ne gardons que les attributs *id* pour les balises <*p*> et <*s*>.

Il est important de noter ici que notre script est assorti d'option en raison des besoins de notre travail ultérieur. Un choix est traité avec l'attribut de *type* et l'autre sans l'attribut de *type* ajouté.

- **Avec l'attribut de type**

Cette version du fichier XML sera utilisée comme corpus de référence dans les travaux ultérieurs. En d'autres termes, dans nos travaux ultérieurs, les résultats obtenus à l'aide des différentes approches de l'identification automatique du discours direct seront comparés au fichier de référence afin d'obtenir la précision des différentes approches. Comme l'annotation sur l'attribut *type* dans le corpus initial a été effectuée manuellement par Barnas (2017) et peut contenir des erreurs, tous les fichiers XML inclus dans le corpus de référence ont été modifiés et vérifiés manuellement. L'attribut de *type* des tokens qui n'appartenaient pas au discours direct (p.ex. dans les incises) a été modifié.

```xml
<p id="p21">
    <s id="s47">
        <t id="t716" num="1" l="—" c="PUNCT" e="" f="" type="dd">—</t>
        <t id="t717" num="2" l="le" c="DET" e="" f="Definite=Def|Number=Sing|PronType=Art" type="dd">L’</t>
        <t id="t718" num="3" l="italien" c="NOUN" e="" f="Gender=Masc|Number=Sing" type="dd">italien</t>
        <t id="t719" num="4" l="?" c="PUNCT" e="" f="" type="dd">?</t>
    </s>
    <s id="s48">
        <t id="t720" num="1" l="se" c="PRON" e="" f="Person=3|PronType=Prs" type="">s’</t>
        <t id="t721" num="2" l="étonner" c="VERB" e="" f="Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin" type="">étonne</t>
        <t id="t722" num="3" l="Vincent" c="PROPN" e="" f="" type="">Vincent</t>
        <t id="t723" num="4" l="." c="PUNCT" e="" f="" type="dd">.</t>
    </s>
</p>
```

Figure 17. Exemple du roman *« Involution »* avec l’attribut *type*

- **Sans l’attribut de *type***

Cette version du fichier XML est traitée comme notre entrée dans la suite de notre travail. Ces fichiers sont principalement utilisés pour l'identification automatique du discours direct par l’approche symbolique. Après avoir écrit les règles pour la reconnaissance automatique du discours direct, les fichiers sans l'attribut *type* sont utilisés comme fichiers d'entrée, et après que les règles ont été traitées, si les tokens appartiennent au discours direct, on leur donne l'attribut *type* et on leur attribue la valeur « *dd* ».

```xml
<p id="p21">
    <s id="s47">
        <t id="t716" num="1" l="—" c="PUNCT" e="" f="">—</t>
        <t id="t717" num="2" l="le" c="DET" e="" f="Definite=Def|Number=Sing|PronType=Art">L’</t>
        <t id="t718" num="3" l="italien" c="NOUN" e="" f="Gender=Masc|Number=Sing">italien</t>
        <t id="t719" num="4" l="?" c="PUNCT" e="" f="">?</t>
    </s>
    <s id="s48">
        <t id="t720" num="1" l="se" c="PRON" e="" f="Person=3|PronType=Prs">s’</t>
        <t id="t721" num="2" l="étonner" c="VERB" e="" f="Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin">étonne</t>
        <t id="t722" num="3" l="Vincent" c="PROPN" e="" f="">Vincent</t>
        <t id="t723" num="4" l="." c="PUNCT" e="" f="">.</t>
    </s>
</p>
```

Figure 18. Exemple du roman *« Involution »* sans l’attribut *type*

2.4. Synthèse des attributs

Afin de mieux comprendre la signification de chaque attribut et son origine, nous les résumons dans le tableau ci-dessous.

			Avec l'attribut de type	Sans l'attribut de type
Nom du attribut	**Signification**	**Origine**		
Id	Nombre de tokens pour le roman Numérotation continue et unique jusqu'à la fin du roman	Stanza Obtenu à partir de la fonction tokenizationen par Stanza	oui	oui
num	Cet attribut indique le numéro du token dans la phrase. Chaque phrase commence par un token de num=1.	Stanza Obtenu à partir de la fonction tokenizationen par Stanza	oui	oui
l	Lemme pour chaque token	Stanza Fonction lemmatization	oui	oui
c	Catégorie UD-POS pour chaque token	Stanza Fonction pos	oui	oui
f	Caractéristiques morphologiques UD-FEAT	Stanza Fonction feats	oui	oui
type	Si ce token est discours direct ou pas	Le corpus initial fourni par Barnas Modifié et vérifié manuellement	oui Si c'est discours direct, le type sera écrit comme 'type = dd', sinon type=""	Non pas d'attribut type,

Tableau 4. Résumé des attributs de niveau <t>

Chapitre 7. APPROCHE SYMBOLIQUE

Dans ce chapitre, nous présentons tout d'abord les principes généraux de notre approche symbolique. Puis, nous détaillons les règles utilisées pour déterminer notre identification automatique du discours direct. Enfin, nous expliquons les grandes lignes du fonctionnement de notre script de reconnaissance.

1. PRINCIPES GENERAUX DE LA METHODES EMPLOYEE

Comme nous l'avons appris dans l'introduction précédente, cette approche pour l'identification automatique du discours direct est exprimée dans un ou plusieurs formalismes spécialisés (des règles, des expressions régulières, etc.), et nous pouvons le faire directement dans un langage de programmation commun.

Lors de la rédaction de règles de travail pour l'identification automatique du discours direct, la première chose à déterminer après le prétraitement est de savoir quels signes de ponctuation sont associés au discours direct dans le corpus. La ponctuation peut être utilisée comme un des indices importants pour la reconnaissance automatique du discours direct. Nous avons donc d'abord décidé de concentrer notre travail sur l'identification des signes de ponctuation associés au discours direct, qui fournit les règles de reconnaissance les plus productives. Pour ce faire, nous observerons et compterons manuellement les signes de ponctuation qui guident le discours direct.

Cette focalisation de notre travail seulement sur la ponctuation reste cependant incomplète. Si nos règles de reconnaissance peuvent identifier efficacement les parties en incise après les parties de discours direct, et marquer l'attribut de type de ces parties en incise comme étant « none », alors nous augmenterons la précision et l'exhaustivité de notre reconnaissance automatique du discours direct. Nous avons ainsi préféré axer notre travail davantage sur le développement de règles pour l'identification automatique du discours direct qui combinent ponctuation et partie incise.

Nous avons en effet pris le parti de considérer que les informations que comporte le corpus initial, après un traitement ultérieur, sont utiles à la tâche d'identification automatique du discours direct. Nous constatons que la reconnaissance de la ponctuation est relativement simple et moins variable, mais que l'identification de l'incise est plus complexe, car elle se présente sous diverses formes et est plus variable. Aussi, nous avons estimé que la bonne

réussite de notre travail résiderait en partie dans la sélection des informations les plus pertinentes pour la reconnaissance d'incise.

Par conséquent, nous pouvons diviser grossièrement l'identification automatique du discours direct à l'aide de l'approche symbolique en deux grandes étapes.

1.1. L'identification automatique préliminaire

Dans cette étape, le discours direct est automatiquement identifié au moyen d'une règle centrée sur la ponctuation. Nous déterminons si le token est le signe de ponctuation que nous recherchons par rapport au discours direct (par exemple, le tiret long, des guillemets français, etc.) en itérant à travers les balises <*t*>. Si la condition pour la reconnaissance automatique de discours direct est remplie, nous marquons tous les tokens guidés par ce signe de ponctuation comme *type="dd"* (Pour le tiret long, généralement le paragraphe entier est marqué comme *type="dd"*, car le tiret long se produise généralement au début de paragraphe. Pour les guillemets français, il s'agit du contenu entre les guillemets français).

1.2. Identifier et retirer la partie incise

Après l'identification automatique préliminaire du discours direct, nous obtenons le résultat que le nombre de marques de discours direct est supérieur au nombre de marques de discours direct exactes. Ce que nous devons faire dans cette étape, c'est examiner plus en détail le résultat initialement étiqueté, c'est-à-dire trouver les parties qui n'appartiennent pas à la partie du discours direct, en d'autres termes la partie d'incise, et supprimer les étiquettes « *type=dd* » de ces parties. Les informations contenues dans la balise <*t*> nous aideront dans l'identification de l'incise. Comme nous l'avons vu dans les chapitres précédents, l'incise est liée à ***verba dicendi*** (verbes introducteurs). Des verbes introducteurs (*dit-il ; cria-t-il ; murmura-t-il…*), placés soit avant la réplique, soit en incise, soit à la fin de la réplique. Les caractéristiques de ces verbes, telles que la conjugaison et le temps, suivent un certain schéma. Ainsi, les attributs fournis dans la balise <*t*>, tels que l'attribut *l*, l'attribut *f*, etc., peuvent servir de base pour déterminer où chercher l'incise.

2. DEFINITION DES REGLES

Dans cette section, nous allons donner une description spécifique des règles utilisées pour trouver les tokens qui appartiennent au discours direct.

Dans le corpus initial (de Magdalena Barnas), nous utilisons un roman pour développer et améliorer les règles de discours direct, que nous désignerons dans la suite par le terme « *corpus de développement* ». Nous avons mentionné au chapitre 5 que le roman sélectionné comme corpus de développement doit éviter ceux dont le discours direct contient trop peu, nous avons donc choisi un roman qui contient un niveau moyen de phrases du discours direct comme corpus de développement, c'est le roman ***« C'est fou ce qu'on voit de choses dans la vie »***. Pour le reste des romans du corpus, c'est-à-dire les 14 romans restants, nous les utilisons comme corpus de test pour tester la précision et l'applicabilité des règles écrites.

Corpus de développement	*« C'est fou ce qu'on voit de choses dans la vie »*
Corpus de test	*« Échine »* *« Etoiles en perdition »* *« Fleur de pavé »* *« Flibustière »* *« Involution »* *« L'enfer »* *« L'oeuvre au noir »* *« La 7e femme »* *« Le zèbre »* *« Bois éternels »* *« Quelqu'un marchait sur ma tombe »* *« Le dragon aux plumes de sang »* *« Le premier jour »* *« La première nuit »*

Tableau 5. Corpus de développement et corpus de tes

2.1. Règle 1 et Règle 2

En analysant et en résumant toutes les phrases de discours direct dans le corpus de développement, nous avons développé deux règles de l'identification automatique du discours direct basées sur la ponctuation et la typographie comme suit. Ceci termine l'étape de l'identification automatique préliminaire du discours direct que nous avons mentionnée ci-dessus.

Nous constatons qu'il existe une caractéristique distinctive des phrases appartenant au discours direct dans le roman : des phrases commencent généralement par tiret long (—), ou apparaissent entre des guillemets français (« »). Ainsi, dans cette étape, nous souhaitons trouver les tokens qui correspondent à cette caractéristique en utilisant les deux règles suivantes et en les marquant avec « *type="dd"* ».

Ex :

30) *— Puisque tu préfères tes bonnes soeurs à ta mère, tu peux repartir chez elles !*[58]

31) *Heureusement, Notre-Dame de Buron me parla : « Arrête tes conneries !» dit-elle d'une voix douce, mais ferme. Je me sauvai, loin de la tentation.*[59]

- **Règle 1 :**

regex = r"^(—.*)$" → Ajout DD (comme "type=dd")

On étiquette avec "type=dd" les tokens compris entre le caractère '—' en début de paragraphe et la fin du paragraphe.

- **Règle 2 :**

regex = r"«(.*?)»" → Ajout DD (comme "type=dd")

On étiquette avec "type=dd" les tokens compris entre "*«*" (le guillemet français ouvrant) et "*»*"(le guillemet français fermant)

Après avoir appliqué ces deux règles, nous obtenons un résultat grossièrement étiqueté. Il marque tous les tokens qui correspondent aux deux règles ci-dessus comme « *type="dd"* ». Cependant, comme nous l'avons mentionné précédemment, ce résultat grossièrement marqué ne traite pas les parties en incise. Par exemple, si la partie en incise

[58] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[59] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

apparaît dans un paragraphe ou une phrase qui correspond aux deux règles ci-dessus, le token appartenant à la partie incise sera toujours marqué comme « *type="dd"* », ce qui n'est évidemment pas ce que nous attendons. Ce que nous devons faire ensuite, c'est comment trouver les tokens qui font partie de l'incise parmi tous les tokens qui ont été marqués comme « *type="dd"* » en établissant une règle et en attribuant à l'attribut type de la balise <*t*> où se trouvent ces tokens la valeur « *type="/"* ».

2.2. Règle 3

En analysant plus en détail les sorties et en passant en revue les tokens qui ne font pas partie du discours direct dans le corpus de développement, nous remarquons que dans les tokens marqués "type=dd", des parenthèses apparaissent parfois. Après observation, nous constatons que ce qui se trouve entre les parenthèses n'appartient pas à la catégorie du discours direct. Par exemple, les crochets (" *[ ]* ") sont utilisés pour des appels de note. Les numéros à l'intérieur sont destinés à faciliter l'explication par l'auteur des parties marquées par des numéros après la fin du roman. Ce qui est entre parenthèses ("*( )*") sert à exprimer la description psychologique du personnage principal, etc. Par conséquent, nous avons créé la Règle 3 pour identifier les tokens qui existent entre les parenthèses ou les crochets et les marquer avec « *type="/"* ».

Ex :

32) *—Revoir la Rote [3].*
Oh !
là là !
les juges ecclésiastiques ne vont pas être contents ![60]

33) *— Mais... heu... il n'est pas un peu jeune ?*
remarquai-je en bégayant.
(Je savais ma mère terriblement sensible aux questions d'âge !)[61]

- **Règle 3 :**

regex_1 = r"\(.*?\)" regex_2 = r"\[.*?\]" → supprimer DD (Transformer type="dd" en type="/")

Une chose à noter à propos de la Règle 3 est que la portée de la Règle 3 est limitée aux tokens dans le résultat grossièrement étiqueté (obtenu en exécutant les Règles 1 et 2)

[60] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[61] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

dont l'attribut *type* a été étiqueté avec « *type="dd"* ». En d'autres termes, nous n'avons pas besoin d'itérer sur parcourir tous les tokens, mais seulement sur travers ceux qui ont été étiquetés avec « *type="dd"* » et déterminer s'ils satisfont à la Règle 3.

Il n'est pas difficile de s'interroger sur la Règle 3 : est-elle très marginale et ne concerne-t-elle que le corpus de développement ? Nous ne pouvons pas répondre à cette question pour le moment, mais afin de garantir la précision de la reconnaissance automatique du discours direct dans le corpus de développement, nous devons conserver la Règle 3. Ceci est dû au fait que les tokens couverts par la Règle 3 ne constituent pas une minorité de cas dans le corpus de développement. Nous pourrons répondre à cette question plus tard dans la partie V. Puisque toutes les règles que nous avons développées pour l'identification automatique du discours direct seront exécutées dans les 14 romans inclus dans le corpus de test, nous pourrons savoir à ce moment-là si la Règle 3 est valable dans d'autres romans différents et s'il s'agit d'une règle marginale.

2.3. Partie en incise

Après les trois règles décrites ci-dessus, nous sommes parvenues à supprimer certains tokens qui n'appartenaient pas au discours direct, comme les tokens entre parenthèses ou entre crochets. Mais il existe encore des tokens qui ne sont pas du discours direct et qui n'ont pas été supprimés, notamment les parties en incise. Par exemple : *dit-il*, etc. Nous constatons que ces tokens d'incise n'ont pas de caractéristiques symboliques non ambiguës (par exemple, ils n'ont pas les deux parenthèses évidentes de part et d'autre). La position de l'incise n'est pas constante ; elle apparaît parfois entre deux discours directs (comme dans l'exemple 34 ci-dessous), et parfois après le discours direct, en guise de fin de phrase (comme dans l'exemple 35 ci-dessous). Ainsi, ce que nous voulons faire dans cette section est de résumer l'occurrence d'incise et de voir s'il existe des modèles, et nous espérons qu'en établissant des règles, nous pourrons parvenir à une reconnaissance automatique de ces tokens.

Ex :

34) *— Hé bé ! dit la prof, je vais peut-être finir par me convertir !*[62]

35) *— Mais ce n'est pas moi qui t'ai envoyé ces roses ! s'écria-t-elle.*[63]

[62] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[63] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

Nous constatons que, même si ces incises n'ont pas de caractéristiques symboliques non ambigües, contrairement aux règles précédentes qui commencent par un long tiret ou sont entre parenthèses ou "«" (le guillemet français ouvrant) et "»" (le guillemet français fermant), l'incise se produise généralement entre deux signes de ponctuation. Par exemple, une virgule, un point, un point d'exclamation ou un point d'interrogation.

Ex :

36) *— Je n'aime pas du tout Céphise, s'exclama Grand-mère.*[64]

37) *— Je m'en fous que ce soit une fille, riposta ma mère, ce que je sais, c'est que j'ai cru mourir de douleur pendant deux jours.*[65]

38) *— Que se passe-t-il ? cria la secrétaire, affolée.*[66]

Après observation, la forme de ces incises peut être résumées comme suit :

(virgule| point | point d'exclamation | point d'interrogation) (Pronom COI ou Réfléchi)? (VerbDic) (tokens*) (virgule| point | point d'exclamation | point d'interrogation)[67]

VerbDic la liste complète des verbes contenus dans VerbDic est présentée en Annexe 3 contient une liste de « *verba dicendi* », les sources de ces verbes étant divisées en deux parties :

1. Une liste de 194 verbes couramment utilisés en incise, telle que résumée dans le rapport de stage de Barnas Magdalena.
2. Sur la base de la liste de verbes fournie par Barnas Magdalena, nous avons également ajouté sept verbes, en fonction de nos observations du corpus (par exemple, "*apprendre*","*avouer*","*rigoler*", etc.).

Il est facile de voir que la ponctuation est très importante pour notre jugement. C'est pourquoi nous avons écrit la Règle 4 suivante pour trouver les signes de ponctuation. Comme mentionné dans la Règle 3 ci-dessus, cette règle ne s'applique pas à tous les tokens du roman, mais aux tokens du résultat qui sont encore étiquetés avec « *type=dd* » dans les balises *<t>* après l'identification automatique du discours direct en utilisant les Règles 1-3.

---

[64] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[65] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[66] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[67] Ici, le ? indique que (Pronom COI ou Réfléchi) cette partie peut ou ne peut pas apparaître. (comme dans les exemples 28 et 29)

♦ **Règle 4 :**

regex = r"\.|,|!|\?(.*?)\.|,|!|\?" → Trouver des incises

On trouve des tokens entre des signes de ponctuation, afin de pouvoir déterminer ultérieurement si ces tokens entre deux signes de ponctuation appartiennent à une incise.

Une fois les lignes de démarcation de début et de fin trouvées, nous devons déterminer les tokens qu'elles contiennent afin de savoir si les tokens entre les deux symboles font bien partie de l'incise. Si les tokens appartiennent à l'incise, nous marquons l'attribut *type* de la balise <*t*> dans laquelle les tokens sont situés à « *type=/* » ; s'ils n'appartiennent pas à l'incise, nous ne changeons pas leur valeur de l'attribut *type*. La question est donc maintenant de savoir comment déterminer si ces tokens font partie de l'incise, et quel type d'information utiliser pour le déterminer.

D'après l'analyse précédente, il est facile de voir que l'accent est mis sur ce qu'est le premier token après le signe de ponctuation (le point de début). Il y a trois types de cas qui peuvent se présenter ici : ***VERB*** ou ***PRON*** ou un ***autre type***.

2.3.1 Autre type de token

Ici, autre type fait référence au fait que le premier token qui apparaît n'est ni ***VERB*** ni ***PRON***. Pour cette information, nous utilisons la valeur fournie par l'attribut *c* dans la balise <*t*>, c'est-à-dire que la valeur de l'attribut c dans la balise <*t*> n'est pas « *c="VERB"* » et « *c= "PRON"* ». Si c'est le cas, nous ne faisons rien. Par exemple, les trois tokens de la phrase ci-dessous « *et nous avec* », seront considérés comme appartenant au discours direct.

Ex :

39) *— D'abord parce que le maréchal Pétain s'est conduit comme un lâche, et nous avec !*[68]

2.3.2 Verbe

Cela fait référence au fait que le premier token qui apparaît après le signe de ponctuation de début est un verbe. C'est-à-dire que pour ce token, la valeur de l'attribut *c* dans la balise <*t*> est « *c="VERB"* ». Ce cas nous intéresse car il correspond déjà en partie à la forme d'occurrence en incise que nous avons décrite, et nous devons maintenant faire

[68] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

d'autres déterminations sur le token correspondant à ce verbe pour savoir s'il s'agit d'un ***VerbDic*** qui satisfait aux exigences.

Un examen plus approfondi des incises dans le *corpus de développement* révèle que tous les verbes ne correspondent pas à nos exigences ; celle-ci requiert la réunion des deux conditions suivantes.

1. Le lemme du verbe

Si le verbe n'appartient pas à la liste ***VerbDic*** que nous avons mentionnée, alors le verbe ne se conformera pas à la forme d'occurrence en incise recherchée. Ici, nous utilisons la valeur fournie par l'attribut *l* dans la balise <*t*>. Les tokens qui apparaissent entre les deux signes de ponctuations (le point de début et le point de la fin) dans ce cas appartiennent au discours direct. Par exemple, ici, « *arrête de râler* » ne sera pas marqué comme « type="/" », car le lemme *arrêter* n'est pas dans notre ***VerbDic***.

Ex :

40) *— Mon trésor, arrête de râler.*[69]

2. La conjugaison du verbe

En français, nous savons que chaque verbe a des conjugaisons différentes selon les situations d'utilisation. En examinant les conjugaisons de verbes qui se produisent en incise, nous constatons que lorsque la conjugaison du verbe est à l'indicatif présent ou à l'indicatif passé ou à l'indicatif imparfait et qu'elle est à la première ou à la troisième personne, le verbe peut répondre au type d'incise que l'on vise. On peut avoir des doutes sur la première personne, car il faut tenir compte du fait que certains romans sont écrits à la première personne.

Pour extraire les informations mentionnées ci-dessus, nous utilisons la valeur fournie par l'attribut *f* de la balise <*t*>. Autrement dit, la valeur de l'attribut *f* doit contenir les informations suivantes :

(Mood = indicatif)+(Person = 3 ou 1)+(Tense = Présent ou Passé Simple ou Imparfait)

Ex :

[69] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

41) — *Je vous en prie, me chuchota la future mère d'une voix suppliante en se levant et en me faisant signe de m'asseoir à sa place, réussissez votre examen !*[70]

La partie « réussissez votre examen » de la phrase d'exemple suivante ne sera pas marquée comme *« type="/ " »* car le verbe *réussissez* est fléchi à la deuxième personne et ne répond donc pas à nos exigences

Lorsque le verbe qui apparaît rempli les deux conditions ci-dessus, nous changeons *« type="dd" »* en *« type="/ " »* pour ces tokens. La partie *« dit-elle »* de l'exemple ci-dessous sera marquée comme *« type="/" »*. C'est parce que « *dit »* satisfait aux deux exigences ci-dessus.

Ex :

42) — *Oui, dit-elle, je te propose un charmant petit cousin, qui adore sortir.*[71]

2.3.3 L'étiquette PRON

Cela fait référence au fait que le premier token qui apparaît après le signe de ponctuation de début est un pronom. Ce cas satisfait également à la forme d'occurrence incise que nous avons résumée. Comme pour le traitement des verbes ci-dessus, nous devons maintenant effectuer une autre détermination du token correspondant à ce pronom pour savoir s'il répond aux exigences.

Pour ce cas, nous avons besoin de deux étapes pour compléter la détermination de celui-ci.

- Étape 1 : person = 3 ou 1

Ici, nous devons d'abord vérifier une contrainte sur l'attribut *f*. En effet, lorsque le token correspondant est *« tu »* ou *« vous »*, même si la valeur de l'attribut *c* dans la balise *<t>* est « *c="PRON"* », il ne remplit pas la condition d'incise. Par exemple, « *vous êtes prêts* » appartient au discours direct.

Ex :

43) — *Mesdemoiselles, vous êtes prêtes ? .*[72]

- Étape 2 : Un token après le ***Pron***

Si l'étape 1 est satisfaite, c'est-à-dire que ce token n'est pas *« tu »* et *« vous »*, nous vérifions une autre condition. D'après notre résumé des formes dans lesquelles l'incise

[70] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[71] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[72] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

apparaît, il est facile de voir que ce n'est que si le pronom est suivi d'un verbe qu'il répond à nos exigences. Nous pouvons donc considérer deux cas.

Cas 1 :

Le token qui suit le ***PRON*** n'est pas un verbe, c'est-à-dire que la valeur de l'attribut *c* dans la balise <*t*> n'est pas « *c="VERB"* ». Nous supposons que les tokens qui apparaissent entre les deux signes de ponctuation (le point de début et le point de la fin) dans ce cas devraient appartenir au discours direct et ne répondent pas à l'exigence d'incise. Nous ne changeons donc pas la valeur de l'attribut *type* pour ces tokens. Les tokens dans la partie de l'exemple « *On n'a presque plus d'argent* » sont toujours marqués avec « *type=" dd"* ». Parce que « *n'* » est pas un verbe ici, il est un ADV.

Ex :

44) *— D'accord, mais où ? On n'a presque plus d'argent…*[73]

Cas 2 :

Le token qui suit le ***Pron*** n'est pas un verbe, c'est-à-dire que la valeur de l'attribut c dans la balise <*t*> est « *c="VERB" »*. Dans ce cas, on vérifie les conditions mentionnées au point 2.3.2 ci-dessus, concernant l'appartenance du verbe à la liste **VerbDic**, le temps et la personne. Ainsi, « *m'expliqua-t-elle froidement* » sera marqué comme "type=/", alors que « *elle va encore en classe* » ne sera pas marqué comme « *type="/"* ». C'est parce que le verbe *aller* ne figure pas dans notre liste de **VerbDic**, même si la conjugaison et sa personne correspondent aux exigences, il ne sera pas marqué comme « *type="/"* ».

Ex :

45) *— Il déteste les enfants, m'expliqua-t-elle froidement.*[74]

46) *— Vous me dérangez pour cette petite jeune fille ? Quoi ? Elle va encore en classe ?*[75]

3. SCRIPT D'IDENTIFICATION AUTOMATIQUE

Le script d'identification automatique du discours direct, dont le pseudo-algorithme est présenté en Annexe 4, a été réalisé en *Python* avec l'aide du module *xml.dom.minidom* pour parser les fichiers du corpus, du module *re* pour déterminer les règles. Il prend en entrée

[73] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[74] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[75] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

des fichiers XML obtenu au chapitre 6 (sans l'attribut *type*) et donne en sortie des nouveaux fichiers XML. Le nouveau fichier XML de sortie ne contient pas seulement les valeurs du fichier d'entrée dans leur intégralité, mais l'attribut *type* sera ajouté à un certain nombre de balises <*t*>. Son fonctionnement peut être résumé, dans les grandes lignes, à un procédé en trois étapes.

3.1. Ajouter l'attribut type

La première étape est la mise en œuvre pour la Règle 1-2. Parcourir tous les tokens du corpus de développement et trouver ceux qui satisfont aux règles, ajouter l'attribut *type* aux balises <*t*> correspondants et leur attribuer une valeur de « *type="dd"* ». Cette étape prend en entrée le fichier XML obtenu au chapitre 6 sans l'attribut *type*, et la sortie prend la forme d'un fichier *.xml*, que nous nommons ***step1_out.xml***.

3.2. Changer l'attribut type de la balise <t> correspondant aux crochets et aux parenthèses

La deuxième étape est la mise en œuvre de la Règle 3. En prenant ***step1_out.xml*** comme entrée, parcourir tous les tokens dans ***step1_out.xml*** qui sont marqués comme « *type="dd"* », et trouver des tokens qui correspondent à la Règle 3, en changeant la valeur de l'attribut *type* des tokens contenus dans ces parties de « *type="dd"* » à « *type="/"* ». La sortie prend la forme d'un fichier *.xml*, que nous nommons ***step2_out.xml***.

3.3. Identifier la partie d'incise

Enfin, la dernière étape consiste en l'identification de la section incise mentionnée ci-dessus. En utilisant ***step2_out.xml*** comme entrée, il parcourt tous les tokens dans ***step2_out.xml*** qui sont marqués comme « *type="dd"* » *et* détermine si les tokens entre les deux signes de ponctuation font partie de l'incise selon les exigences que nous avons mentionnées ci-dessus. S'il s'agit de l'incise, nous changeons la valeur de l'attribut *type* des tokens représentant l'incise de « *type="dd"* » *à* « *type="/"* ». La sortie prend la forme d'un fichier *.xml*, que nous nommons ***step3_out.xml***.

Chapitre 8. APPROCHE NEURONALE

Dans ce chapitre, nous donnons une description détaillée des méthodes d'apprentissage automatique utilisées dans notre travail actuel, l'identification automatique du discours direct. Nous présentons d'abord une brève introduction aux modèle et réseaux neuronaux utilisés dans notre approche d'apprentissage automatique. Ensuite, nous détaillons le traitement des données utilisé pour entraîner notre modèle d'apprentissage automatique. Enfin, nous expliquons le modèle que nous entraînons dans ce cas.

1. INTRODUCTION AUX MODÈLES NEURONAUX

Selon l'introduction précédente (chapitre 4), nous savons qu'il existe de nombreux choix de réseaux neuronaux qui peuvent être utilisés pour l'apprentissage automatique, tels que les CNN, RNN, etc. Depuis quelques années, avec le développement de modèles pré-entrainés sur de larges quantités de données, les modèles Transformers ont connu un grand succès pour de nombreuses tâches en TAL.

1.1. Transformer

Dans l'article « *Attention is all you need* » publié par une équipe de Google en décembre 2017 (Devlin et al. 2017), cet article abandonne complètement les architectures RNN et CNN souvent utilisées dans le passé et propose une nouvelle structure de réseau, le Transformer. Dans un Transformer, chaque mot d'entrée n'est plus isolé, et le vecteur de sortie de chaque mot contient des informations sur les autres mots, grâce à des matrices d'auto-attention. La mesure dans laquelle les mots sont liés les uns aux autres peut être analysée par les poids en sortie de ces matrices, issus d'une fonction *softmax*. Le réseau est entrainé grâce à une simple tâche de complétion : pendant l'entraînement, qui n'est pas surpervisé, on se contente de masquer certains mots des phrases présentées en entrée, et l'on entraine le modèle à retrouver les mots ainsi masqués.

Contrairement au RNN, le Transformer n'a pas le problème de contenir de moins en moins d'informations sur l'entrée précédente à mesure que la distance augmente (problème de disparition de gradient). Quelle que soit la distance spatiale entre le mot actuel et les autres mots, les informations contenues dans les autres mots ne dépendent pas de la distance, mais de l'« attention » que les mots se prêtent mutuellement, ce qui constitue le premier avantage du Transformer. Le deuxième avantage est que le Transformer peut utiliser non seulement les mots précédents mais aussi les mots suivants dans le calcul du mot courant. Ce problème

peut être résolu par un RNN bidirectionnel, mais l'avantage du Transformer est qu'il est intrinsèquement bi-directionnel. Le troisième point est qu'un RNN est une structure séquentielle, qui effectue ses prédictions étape par étape et ne peut les calculer en parallèle. Le Transformer, par son architecture, permet de paralléliser les calculs pour chaque token au niveau de chaque couche, ce qui accélère considérablement les calculs. Enfin, le Transformers intègre des *embeddings* positionnels pour aider le modèle à prendre en compte l'ordre des mots dans la phrase.

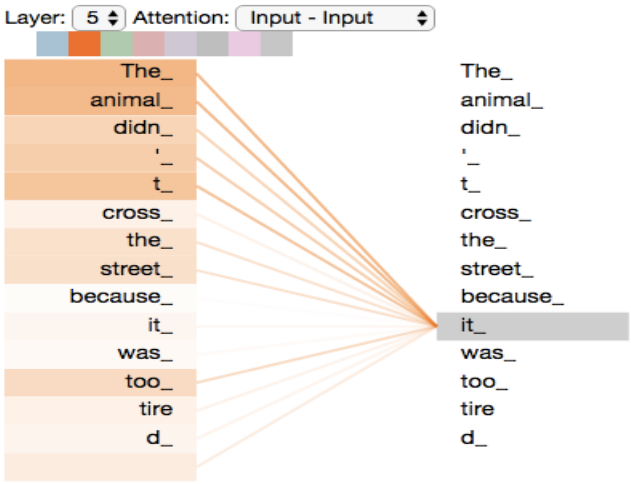


Figure 19. Transformer[76]

1.2. BERT

BERT vient de l'acronyme *Bidirectional Encoder Representations from Transformers*. C'est un modèle de langage développé par Google en 2018 basé sur le Transformer. Le pré-entraînement de BERT peut tirer pleinement parti des informations du contexte, ce qui confère au modèle un pouvoir expressif plus fort. Ceci est bénéfique pour notre tâche de reconnaissance automatique du discours direct. En effet, hormis les informations de ponctuation, notre jugement humain pour déterminer si un texte est un discours direct se base sur le contexte du texte cible. Comme l'ont montré les travaux mentionnés dans l'état de l'art (Byszuk et al., 2020, Kurfali et al., 2020, Brunner et al., 2020), le modèle BERT est un bon choix pour notre travail.

[76] Comme le montre le schéma, il s'agit du poids pour l'un des calculs. Les nuances des lignes reflètent les pondérations, et vous pouvez voir que les deux mots ayant le plus de poids dans *"it"* sont *"the"* et *"animal"*, indiquant que *"it"* est le plus associé à ces deux mots.

2. TRAITEMENT DES DONNÉES

Pour l'apprentissage automatique, nous ne pouvons pas charger directement dans le modèle les données de notre corpus de référence. Nous devons traiter les données dans un format de données que les librairies développées pour ce modèle peuvent analyser.

Nous utilisons ici la librairie *Transformer* fournie par HuggingFace[77] pour implémenter l'entraînement de notre modèle. Pour le format des données, nous devons convertir le fichier *XML* avec attribut type de notre corpus standard dans le format *jsonl* requis. Lors de la conversion au format *jsonl*, nous devons marquer les données en même temps. Comme mentionné précédemment, nous pouvons utiliser différentes étiquettes pour annoter les données. Nous utilisons ici le marquage *BIO tagging*, où chaque token est marqué avec l'une des étiquettes B-I-O suivantes pour une phrase.

- B : *Beginning*, le premier token indiquant le début du discours direct.
- I : *Inside*, indiquant le token appartenant au discours direct.
- O : *Outside*, indique un token qui n'appartient pas au discours direct.

On se base sur la valeur de l'attribut type de chaque token, c'est-à-dire de chaque balise *<t>*, dans le fichier *xml*. Si le *« type=dd »* et qu'il s'agit du premier token de la phrase, alors l'étiquette *« B »* est utilisée. Si *« type=dd »* n'est pas le premier token de la phrase, l'étiquette « I » est utilisée. Le reste des balises *<t>* est marqué par l'étiquette *« O »* (quand la valeur de l'attribut type n'est pas égale à *« type=dd »*)

Ex:

—	Attention	,	Monsieur		Louis	!78		
B	I	I		I	I	I		
—	Je	ne	pense	pas	que	vous	allez	continuer	à
	vous	occuper							
B	I	I	I	I	I	I	I	I	I
	I	I							
	de	bétail	,	dit	l'	architecte	. 79		
	I	I	I	O	O	O	O		

[77] https://huggingface.co/docs/transformers/index
[78] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[79] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

Pour faciliter l'entraînement du modèle par la suite, nous attribuons ici des valeurs numériques aux trois étiquettes *B-I-O*, c'est-à-dire que nous utilisons trois chiffres pour représenter les trois étiquettes.

- B : Désigné par le chiffre 0.
- I : Désigné par le chiffre 1.
- O : Désigné par le chiffre 2.

Ex:

—	Attention	,	Monsieur	Louis	!80	
0	1	1	1	1	1	

—	Je	ne	pense	pas	que	vous	allez	continuer	à
	vous	occuper							
0	1	1	1	1	1	1	1	1	1
	1	1							
	de	bétail	,	dit	l’	architecte	.81		
	1	1	1	2	2	2	2		

Le script pour le format de conversion, dont le pseudo-algorithme est présenté en Annexe 5, a été réalisé en Python avec l'aide du module *xml.dom.minidom* pour parser les fichiers, du modele *json* pour écrire la sortie au format *json*. Il prend en entrée des fichiers XML du corpus standard et donne en sortie des nouveaux fichiers *json*, qui est marqué de 0, 1 et 2 selon la valeur de l'attribut type de la balise <*t*>. Un exemple de résultat est présenté ci-dessous.


```
{"id": "L_enfer#s5643", "tokens": ["Mais", "si", "tu", "ne", "t’", "ennuies", "pas", "et", "si", "tu", "n’", "as", "rien
{"id": "L_enfer#s5644", "tokens": ["Ta", "présence", "me", "fait", "plaisir", "."], "dd_tags": [0, 1, 1, 1, 1, 1]}
{"id": "L_enfer#s5645", "tokens": ["On", "ne", "se", "voit", "plus", "souvent", "."], "dd_tags": [0, 1, 1, 1, 1, 1, 1]}
{"id": "L_enfer#s5646", "tokens": ["J’", "ai", "été", "contente", "quand", "tu", "m’", "as", "dit", "que", "tu", "ne", "
{"id": "L_enfer#s5647", "tokens": ["Discours", "aimable", "et", "sincère", "."], "dd_tags": [2, 2, 2, 2, 2]}
{"id": "L_enfer#s5648", "tokens": ["Elles", "s’", "étaient", "connues", "enfants", "à", "Genève", "dans", "une", "choral
{"id": "L_enfer#s5649", "tokens": ["Annie", "chantait", "toujours", "dans", "une", "chorale", "."], "dd_tags": [2, 2, 2,
{"id": "L_enfer#s5650", "tokens": ["—", "D’", "accord", ",", "dit", "—elle", "."], "dd_tags": [0, 1, 1, 1, 2, 2, 1]}
{"id": "L_enfer#s5651", "tokens": ["Moi", "aussi", ",", "ça", "me", "fait", "plaisir", "."], "dd_tags": [0, 1, 1, 1, 1, 1,
{"id": "L_enfer#s5652", "tokens": ["J’", "aimerais", "tellement", "que", "tout", "s’", "arrange", ",", "pour", "Simon",
{"id": "L_enfer#s5653", "tokens": ["Donc", "elle", "resta", "."], "dd_tags": [2, 2, 2, 2]}
```


Figure 20. Sortie fichier json

[80] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[81] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

3. ENTRAÎNEMENT DU MODÈLE

Après avoir traité les données, nous pouvons passer à l'étape de l'entraînement du modèle. Nous savons tous que l'apprentissage automatique nécessite beaucoup de données pour entrainer des modèles, notamment pour les réseaux neuronaux. Lors de l'entrainement d'un modèle, l'ensemble des données est généralement divisé en différentes parties (ce qu'on appelle le split). Tout d'abord, détaillons de quelle manière nous avons divisé notre ensemble de données.

3.1. Segmentation des ensembles de données

Comme le processus de construction du modèle nécessite également de tester le modèle, de vérifier la configuration du modèle et le degré d'entraînement, surapprentissage ou sous-apprentissage, nous subdivisons les données d'entraînement en deux parties, un ensemble d'entraînement (***Train***) et un ensemble de validation pour le test (***Validation***). Le ***Train*** est utilisé pour entrainer le modèle de réseau neuronal, puis le corpus ***Validation*** est utilisé pour vérifier la validité du modèle et sélectionner les hyper-paramètres (taux d'apprentissage, nombre d'époques, etc.) qui donnent les meilleurs résultats. Enfin, une fois que le modèle a "passé" l'ensemble de validation, nous utilisons alors un troisième corpus **Test** pour évaluer les performances finales du modèle, en évaluant la précision, le taux d'erreur, etc.

Ainsi, notre ensemble de données (les 15 romans contenus dans le corpus standard) est divisé en trois parties : ***Train***, ***Validation*** et ***Test***. Les proportions sont respectivement de 80%, 10% et 10%. Les données de ces 15 romans sont divisées en trois ensembles par un échantillonnage aléatoire uniforme sans intersection.

3.2. Sélection du modèle

La plupart des modèles du modèle BERT ont été formés sur des données anglaises ou sur des données provenant de plusieurs langues, ce qui rend l'application pratique de ces modèles à des langues autres que l'anglais très limitée. Comme notre travail est basé sur des romans français, nous avons voulu utiliser un modèle applicable aux textes français. Nous utilisons donc ici le modèle *CamemBERT*[82], qui est un modèle entraîné sur 138 GB de texte français.

---

[82] Pour l'entraînement du modèle, nous devons utiliser une fonction *TokenizerFast*, qui n'est pas supportée par *FlauBERT*, nous avons donc choisi *CamemBERT* pour notre travail.

Il existe six modèles alternatifs pour *CamemBERT*, et nous avons choisi le modèle à base de CamemBERT le plus couramment utilisé pour ce travail.[83]

3.3. Entraînement du modèle

Une fois tous les préparatifs terminés, nous pouvons commencer l'entraînement de notre modèle. Comme le mentionnent Jannidis et al. (2018) et Kurfali et al. (2020), afin d'éviter d'entraîner un modèle qui ne repose que sur quelques symboles de ponctuation simples et d'espérer permettre au modèle d'apprendre les caractéristiques linguistiques du discours direct, nous avons décidé d'entraîner un modèle en supprimant ces signes de ponctuation. À titre de comparaison, pour voir si la ponctuation joue un rôle important dans la reconnaissance automatique du discours direct, nous avons entraîné deux autres modèles en parallèle : un modèle avec les signes ponctuation et un modèle mixte. Le modèle mixte est un modèle dans lequel 50 % des données d'entrainement sont avec des signes de ponctuation et 50 % sont sans les signes de ponctuation.

Nous savons que nos données d'entraînement proviennent de 15 romans étiquetés manuellement, ce qui ne représente pas une grande quantité de données pour l'entraînement de modèles. Nous avons donc voulu augmenter la quantité de données dans le Train pour voir quel effet cela aurait sur le modèle. Nous ajoutons ici les données des romans français de *Phraseorom*. Nous désignons ce corpus par le terme RAW, car le discours direct y est identifié grâce à une annotation grossière non vérifiée à la main (en utilisant notre *baseline* à base de règles). Au total, RAW contient 30 romans appartenant à des genres fictionnels variés : littérature générale (*GEN*), romans sentimentaux (*SENT*), policiers (*POL*), historiques (*HIST*), de science-fiction (*SF*), fantasy (*FY*). Des informations spécifiques sont données dans le tableau ci-dessous.

Roman	Type	Auteur	Année	La proportion de discours direct (au niveau de token)[84]
« Khanaor 1 Solstice de fer »	FY	Françis Berthelot	1983	22,67%
« Khanaor 2 Equinoxe de cendre »	FY	Françis Berthelot	1986	24,74%

[83] https://huggingface.co/camembert/camembert-base

[84] L'identification automatique de discours direct a été effectuée en utilisant les règles développées par notre méthode symbolique. Les résultats n'ont pas été vérifiés manuellement.

« Avant le déluge »	FY	Raphaël Albert	2018	12,43%
« Le roman de la croix 1 Le coeur de la croix »	HIST	David Camus	2011	31,17%
« Crucifère »	HIST	David Camus	2009	40,72%
« Le roman de la croix 2 Morgennes »	HIST	David Camus	2008	38,96%
« Pour mon plaisir et ma délectation charnelle »	HIST	Pierre Combescot	2009	3,42%
« Les seigneurs de la guerre »	SF	Gérard Klein	1971	29,63%
« Le temps n'a pas d'odeur »	SF	Gérard Klein	1963	32,52%
« Le gambit des étoiles »	SF	Gérard Klein	1958	36,12%
« L'homme de l'espace »	SF	Jimmy Guieu	1981	48,78%
« Nous les martiens »	SF	Jimmy Guieu	1954	30,52%
« Opération aphrodite »	SF	Jimmy Guieu	1981	40,26%
« Hantise sur le monde »	SF	Jimmy Guieu	1953	45,96%
« L'agonie du verre »	SF	Jimmy Guieu	1982	45,53%
« La dimension X »	SF	Jimmy Guieu	1953	45,13%
« Le monde oublié »	SF	Jimmy Guieu	1981	38,04%
« L'invasion de la terre »	SF	Jimmy Guieu	1979	35,51%
« L'univers vivant »	SF	Jimmy Guieu	1986	39,86%

« Au-delà de l'infini »	SF	Jimmy Guieu	1952	45,80%
« Commandos de l'espace »	SF	Jimmy Guieu	1981	41,81%
« Ladies 1 Ladies' taste »	SENT	Laura Trompette	2015	15,92%
« Ladies 2 Ladies' secret »	SENT	Laura Trompette	2015	0,05%
« Le zèbre »	SENT	Alexandre Jardin	1988	10,60%
« Fanfan »	SENT	Alexandre Jardin	1990	13,35%
« Éloge de la phobie »	POL	Brigitte Aubert	2000	1,98%
« Funérarium »	POL	Brigitte Aubert	2006	40,39%
« Pierres de sang »	POL	André Arnaud	1999	62,41%
« Tout seuls »	GEN	Marion Achard	2012	25,21%
« Peine perdue »	GEN	Olivier Adam	2014	7,25%

Tableau 6. Les romans de RAW

Initialement, ces romans sont des fichiers au format XML, similaires au format de notre corpus standard, la seule différence étant l'absence de l'attribut type. Pour résoudre ce problème, nous avons l'intention d'utiliser les règles développées dans notre approche symbolique pour ces romans en les étiquetant avec l'attribut type, ce qui nous permettra de voir si les règles peuvent être utiles pour l'entrainement sans annotation manuelle.

Pour les résultats spécifiques à chaque modèle, nous les présentons un par un dans le chapitre suivant.

Partie 4
-
ANALYSE DES RESULTATS

Chapitre 9. APPROCHE SYMBOLIQUE

1. EVALUATION DE L'APPROCHE SYMBOLIQUE

1.1. Corpus de développement

Roman	***« C'est fou ce qu'on voit de choses dans la vie »***
Nombre de tokens marqués DD dans le corpus de référence	15 170
Nombre de tokens DD à l'issue de l'étape 1	18 049
Nombre de tokens DD à l'issue de l'étape 2	17 358
Nombre de tokens DD à l'issue de l'étape 3	16 215
FN	190
TP	14 980
FP	1235
Rappel	98.75%
Précision	92.38%
F1	95.46%

Tableau 7. Le résultat pour le roman « *C'est fou ce qu'on voit de choses dans la vie* »

Nous avons exécuté les règles que nous avons créées sur le corpus de développement, le roman « ***C'est fou ce qu'on voit de choses dans la vie*** ». Nous avons obtenu les premiers résultats de l'approche symbolique pour la reconnaissance automatique du discours direct pour notre travail.

Dans notre fichier de sortie final, nous avons 16215 tokens marqués comme « *type="dd"* », et pour avoir une meilleure idée de la contribution de chaque règle à l'identification automatique du discours direct, nous résumons le nombre total de tokens marqués comme « dd » dans notre fichier de sortie après chaque étape.

Nous notons également que le nombre de tokens marqués comme « *type="dd"* » dans le corpus grossièrement marqué que nous avons obtenu après l'identification automatique du discours direct avec la Règle 1 et la Règle 2 est beaucoup plus important que notre résultat final, il y a 18 049 de tokens. Avec l'utilisation d'autres règles, le nombre de tokens identifiés automatiquement comme appartenant au discours direct montre une tendance à la baisse. Ces résultats ne sont pas surprenants. En effet, les tokens marqués par la Règle 1 et la Règle 2 ne font rien avec les tokens qui ne font pas partie du discours direct. Par conséquent, le nombre de tokens étiquetés comme « *type="dd"* » devait bien être le plus élevé obtenu. La baisse du nombre de tokens marqués comme « *type="dd"* » aux étapes deux et trois était également attendue : la fonction des Règles 3 et 4 est de trouver les tokens qui ne font pas partie du discours direct et de les marquer comme « *type="/"* ».

Afin de pouvoir comprendre efficacement les résultats finaux de notre reconnaissance automatique du discours direct, nous utilisons les trois valeurs suivantes pour mesurer la précision de notre approche symbolique.

1.1.1 Rappel

Le Rappel est spécifique à notre échantillon original et indique combien d'exemples positifs de l'échantillon original ont été correctement prédits. Il s'agit du nombre de tokens que nous avons correctement identifiés automatiquement dans les tokens qui sont effectivement du discours direct. La formule est la suivante.

$$\text{Recall} = \frac{TP}{TP + FN} \times 100\%$$

1.1.2 Précision

La précision est spécifique à l'identification automatique des résultats du discours direct par notre méthode symbolique, et elle indique combien d'échantillons prédits positifs le sont réellement. C'est-à-dire, combien de tokens qui sont automatiquement identifiés

comme du discours direct par nos règles sont effectivement du discours direct. La meilleure valeur pour la précision est 1 et la pire valeur est 0. La formule est la suivante.

$$Precision = \frac{TP}{TP + FP}$$

1.1.3 F1

Le F1-Score combine la précision et le rappel (c'est la moyenne harmonique), il évalue la capacité d'un modèle de classification à prédire efficacement les individus positifs.

$$F1 = \frac{2 * precision * recall}{precision + recall}$$

D'après ce qui précède, il est facile de voir que pour obtenir ces trois données, nous devons obtenir les trois valeurs TP, FN et FP. Commençons par une brève explication des définitions de ces 3 valeurs.

- TP (*True Positive*) : prédiction correcte d'un échantillon positif. C'est-à-dire qu'un token qui est à l'origine du discours direct est aussi correctement identifié comme du discours direct automatiquement.
- FN (*False Negative*) : prédiction erronée d'échantillons négatifs. C'est-à-dire qu'un token qui est à l'origine du discours direct est incorrectement identifié comme du discours indirect.
- FP (*False Positive*) : prédiction erronée d'échantillons positif. C'est-à-dire qu'un token qui n'est pas du discours direct à l'origine est incorrectement identifié comme du discours direct.

Pour obtenir ces trois valeurs, nous devons utiliser les fichiers du corpus de référence avec l'attribut type que nous avons obtenu au chapitre 5.

Le script permettant d'obtenir les trois valeurs, dont le pseudo-algorithme est présenté en Annexe 5, a été réalisé en *Python* avec l'aide du module *xml.dom.minidom* pour parser les fichiers du corpus. Il prend deux fichiers en entrée, l'un est le fichier de sortie (***step3_out.xml***) obtenu après l'identification automatique du discours direct par notre algorithme symbolique, et l'autre fichier d'entrée est le fichier de référence avec l'attribut

type que nous avons obtenu au chapitre 5. Nous comparons les tokens marqués comme « *type="dd"* » dans ces deux fichiers et obtenons les valeurs *TP*, *FN* et *FP* en sortie.

Au final, pour notre corpus de développement, le roman ***« C'est fou ce qu'on voit de choses dans la vie »***, nous obtenons les résultats suivants. En général, les résultats de l'identification automatique du discours direct sont bons. Nous pouvons voir que le Rappel est assez satisfaisant, à 98,75 %, cela signifie que les règles que nous avons créées font un bon travail d'identification automatique du discours direct. La Précision est inférieure à celle du Rappel, principalement parce que notre valeur de FP n'est pas basse et que 1235 tokens sont incorrectement identifiés comme du discours direct ; cela signifie qu'il y a encore quelques cas qui ne sont pas du discours direct qui ne sont pas couverts par nos règles.

1.2. Corpus de test

1.2.1. Résultats

Ensuite, nous devons appliquer ces règles aux romans de notre corpus de test pour voir si les règles que nous avons créées sont généralisables. Nous avons obtenu les résultats finaux comme indiqué ci-dessous.

Nous pouvons constater que les règles que nous avons créées obtiennent de bons résultats dans l'identification automatique du discours direct, avec une moyenne de 98,62% pour le Rappel, 94,72% pour la précision et 96,56% pour F1. Ces résultats prouvent que les règles que nous avons créées pour identifier automatiquement du discours direct sont efficaces et générales.

Roman	Token_Corpus_Standard	Token_step1_out	Token_step2_out	Supprimer le nombre de patenthèses	Token_step3_out	FN	TP	FP	Rappel	Precision	F1
C'est fou ce qu'on voit de choses dans la vie	15170	18049	17358	691	16215	190	14980	1235	98.75%	92.38%	95.46%
Echine	34633	38090	37989	101	36058	484	34149	1909	98.60%	94.71%	96.61%
Etoiles_en_perdition_BARBET	28576	30159	30133	26	28602	505	28071	531	98.23%	98.14%	98.19%
Involution_HELIOT-SF	18456	19723	19660	63	19224	140	18316	908	99.24%	95.28%	97.22%
Le_zebre_JARDIN	4417	6145	6142	3	5620	25	4392	1228	99.43%	78.15%	87.52%
SENT.GENER.LEVY_premiere nuit	46877	50339	50339	0	47910	321	46556	1354	99.32%	97.17%	98.23%
Flibustière_HELIOT-HIST	21854	24315	24282	33	22793	187	21667	1126	99.14%	95.06%	97.06%
Quelqu_un_marchait_sur_ma_tombe_	17774	19997	19992	5	18276	141	17633	643	99.21%	96.48%	97.83%
L_enfer_BELLETTO	27493	30259	29357	902	28538	432	27061	1477	98.43%	94.82%	96.59%
L'oeuvre au noirYOURCENAR	26477	29833	29720	113	27825	302	26175	1650	98.86%	94.07%	96.41%
Fleur de pavé_LAINE	40504	45678	45662	16	41578	2239	38265	3313	94.47%	92.03%	93.24%
La_7e_femme_MOLAY	39263	41970	41970	0	39715	416	38847	868	98.94%	97.81%	98.37%
ScFi.fr.BORDAGE	36173	38191	38130	61	36899	529	35644	1255	98.54%	96.60%	97.56%
POL.fr.VARGAS.BOIS ETERNELS	76409	82827	82824	3	78002	728	75681	2321	99.05%	97.02%	98.03%
SEN.LEVY.Le_premier_jour	68245	71757	71757	0	68582	494	67751	831	99.28%	98.79%	99.03%
La Moyenne									98.62%	94.72%	96.56%

Figure 21. Les résultats de l'identification automatique du discours direct

1.2.2. Étude de la Règle 3

Concernant la Règle 3, nous pouvons dire qu'il s'agit d'une règle marginale. Dans le corpus de test, il n'y a que dans le roman « ***L'enfer*** » où la Règle 3 élimine un nombre important de tokens, avec 902. Dans les autres romans, les tokens qui répondent aux exigences de la Règle 3 sont soit absents, soit relativement peu nombreux.

Pour vérifier si la Règle 3 a un effet significatif sur les résultats finaux de l'identification automatique du discours direct, nous avons choisi de supprimer la Règle 3, en laissant les autres règles inchangées. La moyenne des résultats de l'identification automatique de discours direct sans la règle 3 est la suivante : F1 est de 96.41%, le Rappel est de 98.63%, la précision est 94.42%.

D'après les résultats, nous pouvons voir que sans la Règle trois, nous avons une baisse des trois valeurs (F1, le Rappel et la précision), mais pas de beaucoup. Nous pouvons conclure que la Règle trois a un rôle à jouer dans l'identification automatique de discours direct, c'est juste que la Règle trois ne joue pas un rôle décisif.

Roman	Token_Corpus_Standard	Token_step1_out	Token_step3_out	FN	TP	FP	Rappel	Precision	F1
C'est fou ce qu'on voit de choses dans la vie	15170	18049	16882	157	15013	1869	98.97%	88.93%	93.68%
Echine	34633	38090	36159	484	34149	2010	98.60%	94.44%	96.48%
Etoiles_en_perdition_BARBET	28576	30159	28628	505	28071	557	98.23%	98.05%	98.14%
Involution_HELIOT-SF	18456	19723	19287	140	18316	971	99.24%	94.97%	97.06%
Le_zebre_JARDIN	4417	6145	5623	25	4392	1231	99.43%	78.11%	87.49%
SENT.GENER.LEVY_premiere nuit	46877	50339	47910	321	46556	1354	99.32%	97.17%	98.23%
Flibustière_HELIOT-HIST	21854	24315	22862	187	21667	1195	99.14%	94.77%	96.91%
Quelqu_un_marchait_sur_ma_tombe_	17774	19997	18281	141	17633	648	99.21%	96.46%	97.81%
L_enfer_BELLETTO	27493	30259	29383	426	27067	2316	98.45%	92.12%	95.18%
L'oeuvre au noirYOURCENAR	26477	29833	27938	278	26199	1739	98.95%	93.78%	96.29%
Fleur de pavé_LAINE	40504	45678	41594	2239	38265	3329	94.47%	92.00%	93.22%
La_7e_femme_MOLAY	39263	41970	39715	416	38847	868	98.94%	97.81%	98.37%
ScFi.fr.BORDAGE	36173	38191	36960	529	35644	1316	98.54%	96.44%	97.48%
POL.fr.VARGAS.BOIS ETERNELS	76409	82827	78005	725	75684	2321	99.05%	97.02%	98.03%
SEN.LEVY.Le_premier_jour	68245	71757	68582	494	67751	831	99.28%	98.79%	99.03%
La Moyenne							98.63%	94.42%	96.41%

Figure 22. Les résultats de l'identification automatique du discours direct sans Règle 3

1.2.3. Analyse de cas particuliers

Il est facile de voir, dans le tableau ci-dessus, que deux des romans ont des résultats inférieurs à la moyenne en matière de l'identification automatique du discours direct : « *Le_zebre* » et « *Fleur de pavé* ». Dans ces deux romans, nous constatons que les valeurs de *Rappel* sont très bonnes, ce qui prouve que nos règles de reconnaissance automatique du discours direct fonctionnent bien. En examinant les résultats, nous constatons que le principal problème se situe au niveau des valeurs *FP*. Les valeurs de *FP* pour les deux romans sont élevées, ce qui entraîne une diminution de leurs valeurs de *Précision* et de *F1*. Les raisons pour lesquelles cela se produit seront données dans la deuxième partie de l'analyse des erreurs.

2. *ANALYSE DES ERREURS*

Après avoir passé en revue les erreurs d'identification qui se produisent dans chaque roman, nous pouvons répartir les erreurs d'identification automatique du discours direct en fonction de différentes causes.

2.1. Erreurs de segmentation de Stanza

Stanza ne sépare pas correctement certains tokens, ce qui conduit à une analyse incorrecte par la suite. La valeur de l'attribut *l* et la valeur de l'attribut *c* dans la balise <*t*> étant affectées, ces valeurs ne correspondent pas aux règles que nous avons créées et font que nos règles identifiées automatiquement du discours direct ne fonctionnent pas.

Ex :

47) — *Est-ce que la Rote a annulé le mariage de vos parents ? demanda Madame R.*[85]

```
<s id="s3578">
    <t id="t49751" num="1" l="demander" c="VERB" e="" f="Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin" type="/">demanda</t>
    <t id="t49752" num="2" l="madame" c="NOUN" e="" f="Gender=Fem|Number=Sing" type="/">Madame</t>
    <t id="t49753" num="3" l="R." c="PROPN" e="" f="" type="dd">R.</t>
</s>
```

Figure 23. Exemple d'erreurs de segmentation par Stanza

Ici, nous pouvons voir une erreur dans la segmentation de *Stanza* pour « *R.* », Ici, c'est « *Madame R* » avec un point, et *Stanza* pense que c'est « *Madame R.* », ce qui entraîne un problème avec les valeurs des attributs *l* et *c*.

<t id="t49753" num="3" l="R." c="PROPN" e="" f="" type="dd">R.</t>[86]

48) — *Je ne comprends pas cette haine de Hanka pour sa petite soeur, gémit-elle.*[87]

```
<t id="t25152" num="14" l="gémit-el" c="PRON" e="" f="Gender=Masc|Number=Sing|Person=3|PronType=Prs" type="dd">gémit-elle</t>
```

Figure 24. Exemple d'erreurs de segmentation par Stanza

Ici, nous pouvons voir que *Stanza* a fait une erreur dans la division de « *gémit-elle* », Ici, qui devrait être de 3 tokens au lieu d'un token, et cause ainsi des problèmes avec les valeurs des attributs *l* et *c*.

---

[85] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[86] Certes le point aurait dû être détaché, mais l'étiquette PROPN est correcte. La règle n'est pas déclenchée du fait de l'absence de ponctuation finale.
[87] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

<t id="t25152" num="14" l="gémit-el" c="PRON" e="" f="Gender=Masc|Number=Sing|Person=3|PronType=Prs" type="dd">gémit-elle</t>

2.2. Erreur d'analyse morphologique de Stanza

Une erreur dans l'analyse des caractéristiques morphologiques de certains tokens a entraîné des informations erronées dans *f* pour correspondre aux règles écrites.

Ex :

49) — *Je veux que vous écoutiez ce que je vais dire, Paulo et vous ! dit calement Lisa.*[88]

```
<t id="t31008" num="1" l="dire" c="VERB" e="" f="Gender=Masc|Number=Sing|Tense=Past|VerbForm=Part" type="dd">dit</t>
```

Figure 25. Exemple d'erreurs d'analyse morphologique par *Stanza*

Ici, la valeur de l'attribut *f* obtenue par l'analyse de *Stanza* est incorrecte ; elle désigne le token « *dit* » comme masculin, singulier, au passé et à la forme participe passé.

Ici « *dit* » ce n'est pas VerbeForm = Part, la valeur de son attribut *f* correct devrait être « ***f="Mod=Ind |Number=Sing|Person=3|Tense=Pres|VerbForm=Fin*** », implique que le token est de l'indicatif présent et est à la troisième personne du singulier.

2.3. Absence de certains verbes de VerbDic

Nous avons constaté, grâce à notre étude des tokens mal étiquetés, que certains tokens sont incorrectement marqués comme discours direct parce qu'une incise n'est pas correctement identifiée. La raison en est que certains lemmes de verbe ne sont pas dans notre ***VerbDic***, et ne peuvent donc pas être mis en correspondance avec nos règles.

Ce que nous pouvons faire pour ce problème est d'enrichir le dictionnaire ***VerbDic*** , ou découvrons s'il existe d'autres ressources pour *verba dicendi*. Si c'est le cas, nous pouvons par la suite remplacer notre ***VerbDic*** par la nouvelle ressource pour voir si nous pouvons améliorer nos résultats de reconnaissance automatique du discours direct.

Ex : *vociférer, soupirer, obstiner,* etc.

2.4. Utilisation d'un temps composé : Avoir + Verbe

Ex :

[88] « Quelqu'un marchait sur ma tombe », Frédéric Dard, 1963

50) — *Oh, eh bien, ma journée est fichue…, avait-elle soupiré.*[89]

Puisque cette situation ne se produit pas dans le corpus développement, il n'y a pas de règle qui permette de juger cette situation. Il existe des solutions à cette situation, on peut rajouter une règle plus complète.

L'émergence de cette question reflète également le problème du développement d'un corpus contenant moins de romans. Comme nous n'avons qu'un seul roman, il est inévitable que nous ne prenions pas en compte toutes les situations. Notre suggestion pour les travaux futurs est d'augmenter le nombre de romans dans le corpus de développement afin de prendre en compte le plus grand nombre de situations possible ou bien de mieux échantillonner.

2.5. Participe Présent

Ex :

51) — *Je bougerai pas d'ici, grommela l'autre, luttant contre lui-même.*[90]

Comme dans le cas 2.4, puisque cela ne se produit pas avec le roman de notre corpus de développement, il n'y a pas de règle qui permette de juger cette situation. Il est possible d'ajouter une règle plus complète pour cette situation.

Pour les travaux futurs, notre recommandation reste d'augmenter le nombre de romans dans le corpus de développement afin de prendre en compte le plus grand nombre de situations possible.

2.6. ADJ / ADV

Certains *ADJ* ou *ADV* apparaissent après l'incise. Parfois, on observe la séquence *DET+NOUN+ADJ*. Nous n'avons pas prévu de règle qui permette de traiter cette situation.

Ex :

52) — *Comment savez-vous tout cela ? me demanda-t-il,visiblement stupéfait.*[91]

53) — *Moi aussi,dis-je, un peu vexée.*[92]

À l'heure actuelle, nous n'avons pas trouvé de règle qui s'applique à cette situation et il est difficile de trouver une règle qui s'y rapporte en se basant uniquement sur les informations fournies dans la balise <*t*> existantes. Nous pouvons donc avoir besoin

[89] « Échine », Philippe Djian, 1988
[90] « Fleur de pavé », Pascal Lainé,1996
[91] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006
[92] « C'est fou ce qu'on voit de choses dans la vie », Nicole de Buron, 2006

d'informations supplémentaires pour nous aider à résoudre ce type de problème. Par exemple, on pourrait résoudre ce type de problème par l'analyse syntaxique en dépendances, en recherchant les modifieurs du verbe ou du sujet.

2.7. Guillemets « »

Comme le roman de notre corpus de développement apparaissant entre " « »" font essentiellement partie du discours direct, nous avons écrit la Règle 2 pour identifier ces cas. Cependant, en testant les romans du corpus, nous avons constaté que tous les tokens apparaissant entre "« »" ne sont pas du discours direct, mais s'applique à des mentions autonymes.

Ex :

54) *En traversant le quartier dit du « Bon Coin », je reconnus, collée sur le panneau de bois d'un café fermé, la petite affiche jaune, comme chaque année, des concerts Hector et Isabel Dioblaníz.*[93]

2.8. Reprise de la narration à l'intérieur du paragraphe

Comme dans l'exemple c-dessous, après un paragraphe commençant par —, on trouve parfois de très nombreuses phrases qui font partie de la narration. À ce stade, les règles que nous avons créées sont incapables d'identifier les tokens qui appartiennent à la phrase de reprise de la narration à l'intérieur du paragraphe.

Nous n'avons pas encore trouvé de solution à cette situation. Mais on peut supposer que le système neuronal entrainé sans ponctuation réussira mieux à traiter ces cas de figure, en s'appuyant sur des indices linguistiques (p.ex. le retour au passé simple).

Ex :

55) *— Was ? demanda son compagnon. Freddy négligea de traduire. Le véhicule venait de se dégager du flot d'ouvriers massés devant l'entrée du tunnel. Une esplanade aux pavés mouillés, luisante et désolée sous la lumière de lampadaires trop hauts, s'offrait. Le conducteur prit à gauche. Les deux gardiens assommés geignaient faiblement sur le plancher de la voiture.*[94]

---

[93] « L'enfer », René Belletto, 1986
[94] « Quelqu'un marchait sur ma tombe », Frédéric Dard, 1963

2.9. Lettre (discours épistolaire)

C'est la principale raison des résultats insatisfaisants de l'identification automatique du roman « *Le_zebre* ». Dans « *Le_zebre* », on trouve un grand nombre de paragraphes indiquant le contenu d'une lettre. Comme le soulignent Schöch et al. (2016), la reconnaissance des lettres est un domaine difficile. Les lettres qui apparaissent dans le roman sont partiellement non identifiables. La lettre commence par "«"et ne comporte pas de symbole " »". Selon les règles, tous les tokens après "«" sont marqués comme « *type="dd"* ». Et les lettres sont décrites à la première personne, ce qui rend plus difficile l'identification automatique du discours direct.

Il n'existe aucun moyen simple de traiter l'émergence de tels problèmes à l'aide d'une approche basée sur des règles, et l'on espère qu'une solution à l'émergence de tels problèmes pourra être apportée plus tard dans l'apprentissage automatique.

Ex :

56) *<p467>*
Dans l'entrée, le Zèbre trouva une lettre de Camille, posée sur le carrelage.
</p>
< p468>
« Gaspard,
</p>

< p469>

« Je te quitte parce que je t'ai compris.

Je te quitte pour que nous ne devenions jamais un vieux couple.

Je te quitte par amour, avant que nos sentiments se figent en habitudes.

Je te quitte comme on sort du cinéma pour ne pas voir mourir le héros.

Je te quitte comme on refuse de voir la dépouille des défunts qu'on a aimés pour conserver l'image de leur regard lumineux.

Je te quitte parce que l'amour est poète et que les poètes meurent jeunes.

Je te quitte parce que Roméo et Juliette ne peuvent fêter leurs noces d'argent.

Je te quitte pour te garder tel qu'en ta beauté folle de cette nuit où tu m'es apparu dans une cage d'escalier, vêtu d'une serviette.

</p>[95]

2.10. Composition de romans

Certains romans ont leur propre typographie et ne suivent pas le format typographique habituel, de sorte que les règles n'identifient pas correctement ces tokens. Il s'agit d'un facteur important influençant les résultats inférieurs à la moyenne de l'identification automatique du discours direct du romans « *Fleur de pav*é ».

Pour le roman « *Fleur de pavé* », il y a des discours directs dans le roman, qui commencent généralement par tiret long (—), mais se termine par un guillemet français fermant.

Dans ce cas, les règles n'ont aucun moyen d'identifier la partie « *ricana Basiloff* ». En effet, le guillemet français fermant n’est pas utilisé comme ponctuation d'identification à la Règle 4. Dans ce cas, nous pouvons probablement résoudre ce problème en ajoutant une règle plus complète.

Ex :

57) *— Tu parles ! » ricana Basiloff.* [96]

[95] « Le_zebre », Alexandre Jardin, 1988
[96] « Fleur de pavé », Pascal Lainé,1996

Chapitre 10. APPROCHE NEURONALE

Ce chapitre se concentre sur les résultats finaux des différents modèles utilisés pour identifier automatiquement du discours direct à l'aide d'une approche d'apprentissage automatique.

1. EVALUATION DE L'APPROCHE NEURONALE

Comme nous l'avons mentionné au chapitre 8, nous avons trois modèles principaux : *modèle avec symbole* (avec les signes de ponctuation), *modèle sans symbole* (sans les signes de ponctuation) et *modèle mixte*. Nous commençons par présenter les résultats pour chacun de ces trois modèles.

	Modèle_Avec_symbole	Modèle_Sans_symbole	Modèle_Mixte_symbole
Training Loss	0.1067	0.0875	0.0605
Validation Loss	0.2583	0.2642	0.3576
Precision (obtenus sur le corpus de validation)	0.8906	0.8856	0.8788
Recall (obtenus sur le corpus de validation)	0.9204	0.9176	0.8986
F1 (obtenus sur le corpus de validation)	0.9053	0.9013	0.8886
Accuracy (obtenus sur le corpus de validation)	0.9319	0.9364	0.9230

Figure 26. Les résultats de trois Modèles

1.1. Modèle avec symbole

Dans ce modèle, nous ne modifions pas les paramètres tels que l'époque, le taux d'apprentissage, etc. Nous choisissons les hyperparamètres en utilisant l'optimiseur.

Nous devons ici mentionner une métrique : Accuracy. Il indique la valeur de la quantité de données correctement prédites par rapport à la quantité totale de données impliquées dans la prédiction. Il faut noter que l'Accuracy n'est pas nécessairement très pertinente ici, car elle est parfois très élevée pour les corpus où on a peu de discours direct à découvrir.

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Pour ce modèle, nous sélectionnons un taux d'apprentissage de *2e-05* et entraînons 3 époques, et nous pouvons voir d'après les résultats que la reconnaissance automatique du discours direct est assez efficace. Accuracy est de 93,19 % et F1 de 90,53 %. 92,04 % et 89,06 % pour le rappel et la précision.

1.2. Modèle sans symbole

Dans ce modèle, nous modifions le paramètre « *epoch* » et constatons que les résultats sont meilleurs lorsque « *epoch=4*» que «*epoch=3*» [97]. Notre taux d'apprentissage n'a pas changé et est de 2e-05. Nous finissons par obtenir les résultats suivants : Accuracy à 93,64% et F1 à 90,13%. Le rappel et la précision sont respectivement de 91,76% et 88,56%.

1.3. Modèle mixte

Dans ce modèle, nous changeons le paramètre *epoch*, nous choisissons « *epoch=5* ». Notre taux d'apprentissage reste le même, *2e-05*. Nous obtenons les résultats suivants : Accuracy à 92.30%, F1 à 88.86%. Le rappel et la précision sont respectivement 89.86% et 87.88%.

1.4. Modèle Train+RAW[98]

Les romans du corpus RAW proviennent de *Phraseorom*, et nous les avons divisés en trois sous-corpus, RAW10, RAW15 et RAW20, chacun contenant respectivement 1 262 172, 1 834 048 et 2 316 273 tokens (pour 1 million, 1,5 million et 2 millions de tokens supplémentaires). Nous ajoutons chacun de ces trois sous-corpus aux données du TRAIN pour obtenir nos trois nouveaux modèles, *Train_RAW10, Train_RAW15* et *Train_RAW20*.

Pour ces trois modèles, nous choisissons *epoch=2* et le taux d'apprentissage est de *2e-05*. Les résultats finaux obtenus sont les suivants.

L'Accuracy de *Train+RAW10* est de 93,57 % et le F1 de 88,97 %. Le rappel et la précision sont respectivement de 90,56 % et 87,44 %.

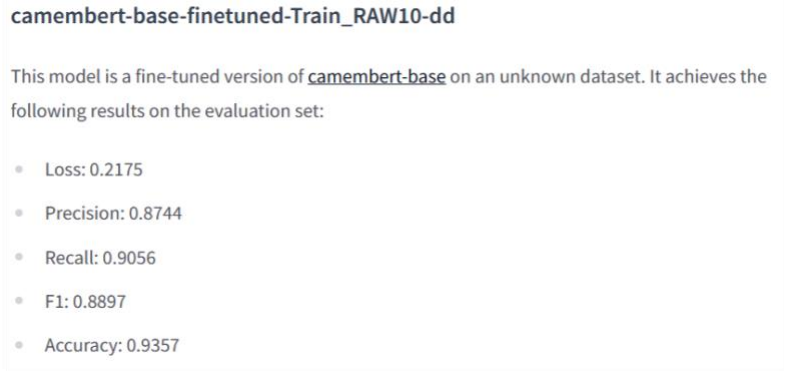


Figure 27. Train_RAW10

[97] Après avoir sélectionné les paramètres à l'aide de l'optimiseur, nous constatons que la valeur de F1 augmente toujours. Nous choisissons donc de modifier l'époque et de nous entraîner davantage pour obtenir la meilleure valeur de F1.
[98] On utilise la stratégie "avec symbole" pour ces modèles

L’Accuracy du modèle *Train_RAW15* est de 92,56 % et le F1est de 88,97 %. 90,09 % et 87,88 % pour le rappel et la précision respectivement.

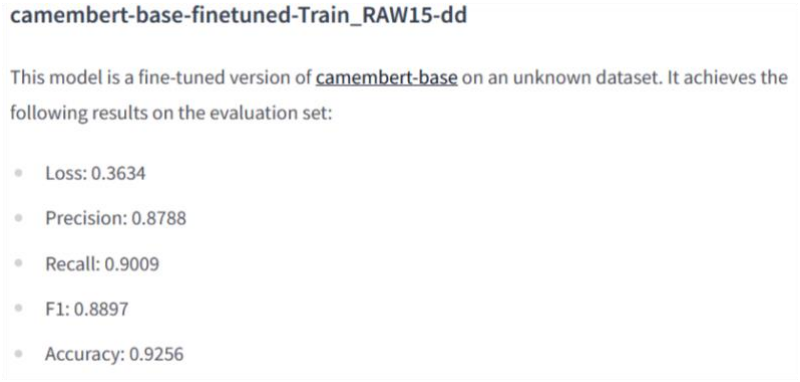


Figure 28. Train_RAW15

L'Accuracy du modèle *Train_RAW20* est de 92,09% et le F1 de 87,79%. Le rappel et la précision sont respectivement de 89,00% et 86,61%.

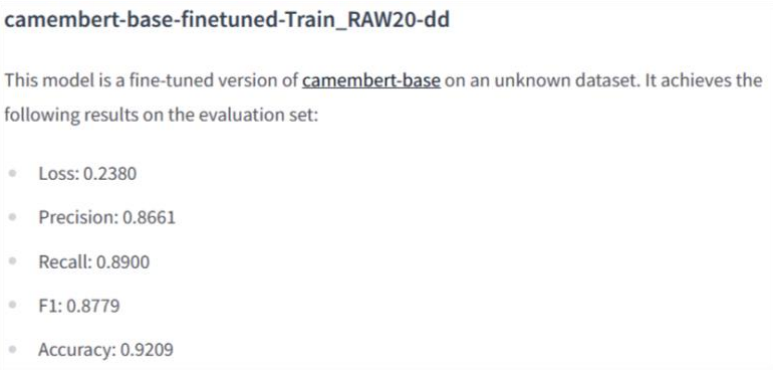


Figure 29. Train_RAW20

1.5. Modèle RAW

Afin d'analyser l'effet de l'ajout du corpus *RAW* sur le modèle, nous avons également choisi d'entraîner trois modèles contenant uniquement des données *RAW*. Il s'agit de *RAW10, RAW20* et *RAW30*. Le nombre de tokens inclus est respectivement de 1 246 391, 2 818 267 et 3 300 492. Pour chaque corpus RAW, la proportion de discours direct à découvrir est 26.78%, 19.81% et 20.09% (au niveau de token).

Les résultats de ces trois modèles peuvent être comparés aux modèles de la section 1.4 pour voir comment l'inclusion de *RAW* affecte les performances des modèles.

Pour les trois modèles de *RAW*, nous avons modifié le paramètre epoch, nous avons choisi *epoch=1*. Nous avons constaté que les meilleurs résultats étaient obtenus avec *epoch=1*. Notre taux d'apprentissage n'a pas changé, il est toujours de *2e-05*.

Pour *RAW10*, nous avons obtenu les résultats suivants : 99,35% d'Accuracy et 87,67% de F1. Le rappel et la précision étaient respectivement de 87,02% et 88,33%. *RAW10* contient la plus grande proportion de discours direct, mais les données dans *RAW* n'ont pas été modifiées manuellement, donc plus la proportion de discours direct incluse est élevée, plus le taux d'erreur des données est élevé. Il en résulte une grande différence dans les valeurs d'Accuracy et de F1.

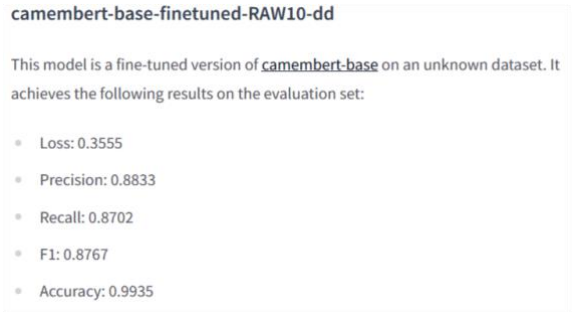


Figure 30. RAW10

Pour *RAW15*, nous obtenons les résultats suivants : accuracy à 99,39 % et F1 à 88,01 %. Le rappel et la précision sont respectivement de 87,61 % et 88,42 %.

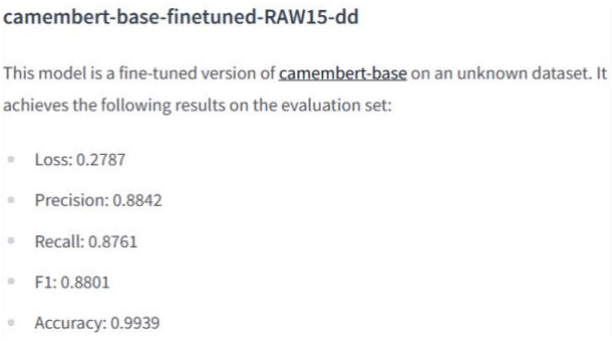


Figure 31. RAW15

Dans le cas de *RAW20*, nous avons obtenu les résultats suivants : accurcy à 99,26 % et F1 à 85,65 %. Le rappel et la précision étaient respectivement de 84,29 % et 87,06 %.

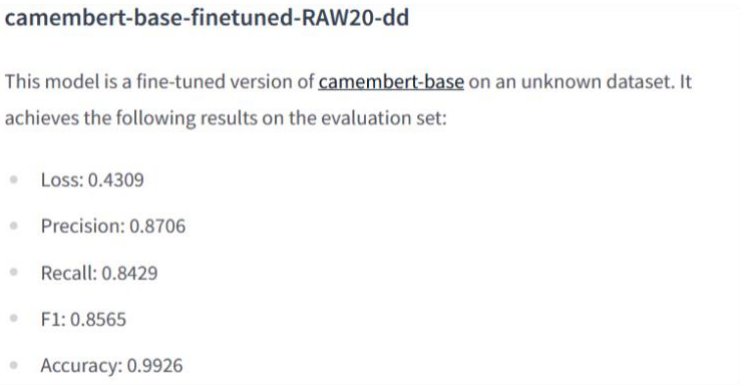


Figure 32. RAW20

	RAW10	RAW15	RAW20
Training Loss	0.1116	0.1806	0.1360
Validation Loss	0.3555	0.2787	0.4309
Precision (obtenus sur le corpus de validation)	0.8833	0.8842	0.8706
Recall (obtenus sur le corpus de validation)	0.8702	0.8761	0.8429
F1 (obtenus sur le corpus de validation)	0.8767	0.8801	0.8565
Accuracy (obtenus sur le corpus de validation)	0.9935	0.9939	0.9926

Figure 33. Les résultats de trois *RAW Modèles*

2. EVALUATION DES RÉSULTATS

Les résultats spécifiques de l'entrainement pour chaque modèle ont été détaillés ci-dessus, et dans l'ensemble, il semble que les résultats de nos trois principaux modèles soient similaires. Cependant, à travers le fichier de *Validation*, nous pouvons encore observer que le *modèle avec symboles* est le plus performant, suivi par le *modèle sans symboles*, et enfin par le *modèle mixte* (sur la base de la valeur de F1). Ce résultat n'est pas surprenant, car comme nous l'avons dit précédemment, les signes de ponctuation sont un marqueur important pour identifier le discours direct. De meilleurs résultats sont obtenus dans les modèles qui incluent des signes de ponctuation.

2.1. Trois modèles principaux

Maintenant, afin d'évaluer les performances des modèles de manière plus généralisable, nous devons utiliser nos fichiers de « *Test* » (et non de Validation, pour lesquels on a optimisé les hyperparamètres). En plus d'utiliser les fichiers de Test qui accompagnent chaque modèle, nous utilisons également différents fichiers de test afin d'évaluer les modèles. Pour chaque modèle, nous avons utilisé 3 fichiers de test. Il s'agissait du fichier de « *test avec symboles* », du fichier de « *test sans symboles* » et du fichier de « *test avec symboles mixtes* ». Les résultats spécifiques sont présentés ci-dessous.

Nous pouvons clairement voir que le modèle obtient les meilleures performances dans le fichier de test correspondant au cas de figure pour lequel il a été entrainé.[99] (En jaune). Globalement, le *modèle mixte* donne les meilleurs résultats, et il est plus robuste, car il donne un bon résultat que le fichier de *test* contienne ou non des symboles. Le fichier de *test avec symboles* a un F1 de 91,54% et le fichier de *test sans symboles* a un F1 de 90,16%. Le *modèle sans symboles* présente des résultats inférieurs et semble donc moins robuste. Comme nous pouvons le constater lorsque le fichier de test contient de la ponctuation, la valeur F1 du

[99] Le modèle est avec symbole, son fichier de test compagnon est le fichier de test avec symbole, et ainsi de suite

modèle sans symboles chute à 76,21 %. Nous pouvons en conclure que les signes de ponctuation ont un grand impact sur l'identification automatique du discours direct et qu'ils fournissent des informations irremplaçables pour l'identification automatique du discours direct.

	Modèle_Avec_symbole	Modèle_Sans_symbole	Modèle_Mixte_symbole
Train			
Precision	95.1835%	95.3203%	96.8097%
Recall	96.5088%	96.7895%	97.6209%
F1	95.8415%	96.0493%	97.2136%
Accuracy	97.9350%	98.4830%	99.1138%
Validation			
Precision	89.0632%	88.5609%	87.8804%
Recall	92.0434%	91.7559%	89.8572%
F1	90.5288%	90.1301%	88.8578%
Accuracy	93.1881%	93.6399%	92.3025%
Test_avec_symbole			
Precision	90.0894%	78.4464%	90.3991%
Recall	93.2085%	74.1029%	92.7180%
F1	91.6224%	76.2128%	91.5438%
Accuracy	94.9200%	87.9157%	95.1278%
Test_sans_symbole			
Precision	87.2410%	88.8622%	89.1875%
Recall	88.7642%	92.0221%	91.1562%
F1	87.9960%	90.4146%	90.1611%
Accuracy	93.6929%	94.5159%	94.3757%
Test_mixte_symbole			
Precision	88.8706%	84.1597%	90.2734%
Recall	91.3852%	83.7815%	92.7263%
F1	90.1104%	83.9702%	91.4834%
Accuracy	94.5416%	91.3525%	95.1252%

Figure 34. Trois modèles principaux

2.2. Avec RAW

Nous pouvons constater à partir des résultats que l'ajout du corpus *RAW* n'améliore pas la précision du modèle (les comparer avec la stratégie “avec_symbole”). Relativement parlant, le modèle est plus performant avec *Train_RAW15*, et les résultats sont proches de ceux du *Modele_avec_symbole*, quoique légèrement dégradés par rapport à celui-ci.

	Modèle_Avec_symbole	Modèle_Sans_symbole	Modèle_Mixte_symbole	Modèle_Train+RAW10	Modèle_Train+RAW15	Modèle_Train+RAW20
Train						
Precision	95.1835%	95.3203%	96.8097%	90.4263%	95.6125%	88.6471%
Recall	96.5088%	96.7895%	97.6209%	92.9174%	96.4453%	91.3293%
F1	95.8415%	96.0493%	97.2136%	91.6549%	96.0271%	89.9682%
Accuracy	97.9350%	98.4830%	99.1138%	95.9833%	98.5037%	95.0120%
Validation						
Precision	89.0632%	88.5609%	87.8804%	87.4353%	87.8812%	86.6132%
Recall	92.0434%	91.7559%	89.8572%	90.5593%	90.0852%	88.9996%
F1	90.5288%	90.1301%	88.8578%	88.9699%	88.9696%	87.7902%
Accuracy	93.1881%	93.6399%	92.3025%	93.5675%	92.5616%	92.0887%
Test_avec_symbole						
Precision	90.0894%	78.4464%	90.3991%	88.1958%	89.3813%	87.6603%
Recall	93.2085%	74.1029%	92.7180%	91.1690%	91.0289%	90.0687%
F1	91.6224%	76.2128%	91.5438%	89.6578%	90.1976%	88.8482%
Accuracy	94.9200%	87.9157%	95.1278%	94.6625%	94.3233%	94.0304%
Test_sans_symbole						
Precision	87.2410%	88.8622%	89.1875%	83.2169%	84.5437%	82.3794%
Recall	88.7642%	92.0221%	91.1562%	84.5695%	83.7036%	81.9292%
F1	87.9960%	90.4146%	90.1611%	83.8878%	84.1215%	82.1536%
Accuracy	93.6929%	94.5159%	94.3757%	92.9990%	92.3968%	92.0103%
Test_mixte_symbole						
Precision	88.8706%	84.1597%	90.2734%	86.2833%	87.4224%	85.4184%
Recall	91.3852%	83.7815%	92.7263%	88.5066%	87.9871%	86.5759%
F1	90.1104%	83.9702%	91.4834%	87.3808%	87.7038%	85.9932%
Accuracy	94.5416%	91.3525%	95.1252%	94.1596%	93.7176%	93.3531%

Figure 35. Modèles Train avec RAW

Pour pouvoir trouver la raison pour laquelle l'augmentation de la quantité de données *TRAIN* n'a pas amélioré les performances du modèle, nous avons entraîné trois

modèles *RAW* distincts. Nous avons observé que dans l'ensemble, pour le fichier de test *test_avec_symbole*, le F1 des trois modèles était respectivement de 87,68%, 87,93% et 87,03%, sans différence significative entre les trois modèles. D'après le reste des fichiers de test, il semble que le modèle *RAW15* soit plus performant que les deux autres modèles.

	RAW10	RAW15	RAW20
Train			
Precision	94.4015%	91.5391%	92.6288%
Recall	94.8490%	92.1585%	93.4388%
F1	94.6247%	91.8477%	93.0321%
Accuracy	99.6793%	99.5446%	99.6148%
Validation			
Precision	88.3331%	88.4157%	87.0585%
Recall	87.0211%	87.6076%	84.2867%
F1	87.6722%	88.0098%	85.6502%
Accuracy	99.3543%	99.3912%	99.2579%
Test_avec_symbole			
Precision	88.2237%	88.0733%	88.0498%
Recall	87.1363%	87.7916%	86.0442%
F1	87.6766%	87.9322%	87.0354%
Accuracy	99.3674%	99.3835%	99.3316%
Test_sans_symbole			
Precision	82.0400%	82.5329%	80.9260%
Recall	78.4511%	80.4762%	76.1886%
F1	80.2054%	81.4916%	78.4859%
Accuracy	98.9832%	99.0608%	98.8996%
Test_mixte_symbole			
Precision	86.1311%	86.3833%	85.6370%
Recall	84.2521%	85.8159%	82.8915%
F1	85.1812%	86.0987%	84.2419%
Accuracy	99.2490%	99.3040%	99.2005%

Figure 36. Modèles RAW

Nous pouvons en conclure que l'exactitude des données utilisées pour entraîner les modèles est plus importante que la quantité de données. Comme les données de la base *RAW* ajoutée n'ont pas été corrigées manuellement, les résultats obtenus pour les trois modèles ne sont pas aussi bons que les résultats précédents pour le modèle principal. Ainsi, l'ajout du corpus *RAW* au fichier d'entraînement et l'augmentation des données d'entraînement n'améliorent pas les résultats des modèles.

2.3. Lettre

Dans notre approche symbolique, il y a une question qui nous préoccupe - l'identification des lettres. Pour cette partie du contenu, il n'y a pas de bon moyen pour nous de la séparer du discours direct avec des règles. Afin de voir si le modèle d'apprentissage automatique entraîné peut résoudre ce problème efficacement, nous avons créé un document séparé pour le fichier de test qui appartient au roman « *Le_zebre* », que nous appelons "*Le_Zebre*". Le fichier "*Le_Zebre*" contient la partie des lettres et les autres parties. Le fichier "*Le_Zebre*" a également été désymbolisé et nommé "*Le_Zebre_sans_symbole*".

2.3.1. *Trois modèles principaux*

D'après le graphique, nous pouvons voir que lorsque « *Le_Zebre* » a les signes de ponctuation, le modèle mixte symbole obtient la meilleure reconnaissance, à 92.10%. Le modèle avec symbole présente le moins bon résultat de reconnaissance, avec 87,88 %. Afin de comparer les résultats des deux méthodes pour la reconnaissance automatique du discours direct, nous avons identifié les mêmes parties du fichier « *Le_Zebre* » à l'aide de nos règles et calculé les résultats (les résultats sont présentés dans la figure *Règle_Résultat*). D'après les résultats, nous pouvons voir que les trois modèles entraînés par l'apprentissage automatique surpassent l'approche symbolique. Et les résultats des modèles sans symbole sont en général meilleurs que ceux des modèles avec symbole, notamment au niveau de la précision. Ces résultats suggèrent que les modèles sans symbole gèrent mieux le bruit engendré par la présence de lettre, et sont donc plus précis.

	Modèle_Avec_symbole	Modèle_Sans_symbole	Modèle_Mixte_symbole	Règle_Résultat
Le_Zebre				
Precision	84.0782%	89.4410%	89.8256%	57.2062%
Recall	92.0489%	88.0734%	94.4954%	/
F1	87.8832%	88.7519%	92.1013%	72.7786%
Accuracy	96.6020%	95.8874%	96.9520%	/
Le_Zebre_sans_symbole				
Precision	81.6384%	90.6706%	86.0399%	/
Recall	88.9230%	95.6923%	92.9231%	/
F1	85.1251%	93.1138%	89.3491%	/
Accuracy	95.1586%	98.0401%	96.4312%	/

Figure 37. Résultats pour lettre de 3 modèles principaux

2.3.2. *Modèles Train+RAW*

Nous pouvons voir sur la figure que la performance du modèle après l'ajout du corpus *RAW* est de 88,01%, 87,97% et 87,00% pour « *Le_Zebre* » respectivement, ce qui ne constitue pas une amélioration significative de la performance du modèle. D'après les résultats, nous pouvons voir que les trois modèles entraînés par l'apprentissage automatique surpassent l'approche symbolique. Et les résultats de la reconnaissance de document avec les signes de ponctuation (« *Le_Zebre* ») sont généralement meilleurs que ceux du document sans les signes de ponctuation (« *Le_Zebre_sans_symbole* »).

	Modèle_Avec_symbole	Modèle_Sans_symbole	Modèle_Mixte_symbole	Règle_Résultat	Modèle_Train+RAW10	Modèle_Train+RAW15	Modèle_Train+RAW20
Le_Zebre							
Precision	84.0782%	89.4410%	89.8256%	57.2062%	84.3137%	84.5070%	83.2402%
Recall	92.0489%	88.0734%	94.4954%	/	92.0489%	91.7431%	91.1315%
F1	87.8832%	88.7519%	92.1013%	72.7786%	88.0117%	87.9765%	87.0073%
Accuracy	96.6020%	95.8874%	96.9520%	/	96.5291%	96.3103%	96.1062%
Le_Zebre_sans_symbole							
Precision	81.6384%	90.6706%	86.0399%	/	80.8333%	82.1530%	80.7365%
Recall	88.9230%	95.6923%	92.9231%	/	89.5385%	89.2308%	87.6923%
F1	85.1251%	93.1138%	89.3491%	/	84.9635%	85.5457%	84.0708%
Accuracy	95.1586%	98.0401%	96.4312%	/	95.8900%	95.4073%	94.6175%

Figure 38. Résultats pour lettre de modèle *Train+RAW*

2.4. Conclusion

D’après l'analyse ci-dessus, nous pouvons conclure que l'augmentation de la quantité de données *TRAIN* a plutôt dégradé les performances du modèle. C'est sans doute dû au fait que les données *RAW* ne sont pas corrigées manuellement et sont donc sujettes à des erreurs. Ces données erronées peuvent interférer avec l'apprentissage des modèles.

Nous ne pouvons pas répondre à la question de savoir si la solution optimale a été atteinte pour chaque modèle. Comme nous n'avons ajusté que le paramètre *epoch*, *epoch* représente le nombre de tours de formation du modèle. Une *epoch* signifie que chaque échantillon de l'ensemble de données d'apprentissage a la possibilité de mettre à jour les paramètres du modèle interne. Nous espérons qu'en modifiant *epoch*, le modèle sera capable de mieux apprendre chaque échantillon. Dans les travaux futurs, nous pouvons ajuster d'autres paramètres pour voir si nous pouvons obtenir de meilleurs résultats. Dans l'ensemble, le *modèle mixte symbole* fonctionne le mieux pour l’identification automatique du discours direct et les travaux futurs pourraient se concentrer sur ce modèle.

Dans le cas de la lettre que nous avons mentionnée précédemment, le système nerveux est plus performant lorsqu'il y a un problème avec le système symbolique. Les résultats de la reconnaissance automatique de discours direct pour la partie lettre sont bien meilleurs que ceux de l'approche symbolique, que le modèle inclue ou non la signe de ponctuation. Toutefois, nous devons admettre que le modèle symbolique, dans l'ensemble, est plus performant (96.56% de F1 moyenne).

Conclusion

Le travail que nous présentons montre qu'il est possible d'effectuer une reconnaissance automatique du discours direct dans les romans français en se basant sur des méthodes symboliques et des méthodes d'apprentissage automatique.

Tout d'abord, il était nécessaire d'identifier les objets linguistiques sur lesquels notre étude allait se concentrer. Pour ce faire, nous avons d'abord dû faire un résumé des phénomènes qui sont considérés comme faisant partie du discours direct. Nous avons démontré que cette tâche n'est pas triviale.

Nous avons ensuite mis en œuvre une approche symbolique pour l'identification, en adoptant une démarche similaire à celle de Sini et al. (2018). Les règles de reconnaissance que nous avons élaborées se concentrent d'abord sur les informations que les signes de ponctuation nous apportent. Les signes de ponctuation peuvent en général nous indiquer une bonne partie des occurrences de discours direct. Sur cette base, l'accent mis sur les verbes introducteurs du discours direct facilite notre capacité à trouver des tokens qui sont mal étiquetés en tant que parties du discours direct, comme les parties en incise, ce qui a conduit à des résultats plus précis pour l'identification automatique du discours direct.

Deuxièmement, pour évaluer l'approche basée sur l'apprentissage automatique, nous avons entrainé des modèles en tenant compte de l'impact de ces signes de ponctuation en tant que facteur sur la reconnaissance automatique du discours direct. Ainsi, lors de notre entraînement du modèle, contrairement aux travaux de Byszuk et al. (2020), Kurfali et al. (2020) et d'autres, nous n'avons pas choisi singulièrement de former un modèle avec ou sans signes de ponctuation. Nous avons choisi de faire un entraînement de 3 modèles : avec symboles, sans symboles et symboles mixtes, afin que nous puissions à la fois permettre au modèle d'apprendre les caractéristiques linguistiques du discours direct plutôt que la typographie (modèle sans symbole) et observer l'importance des informations liées aux signes de ponctuation dans diverses situations. Nous avons ensuite testé les performances des trois modèles en utilisant le même fichier de test. Les résultats obtenus montrent que le modèle symbole mixte est le plus robuste. Cela est dû au fait que le modèle apprend à la fois les informations fournies par les signes de ponctuation et les caractéristiques linguistiques du discours direct.

Par ailleurs nous avons testé l'effet de la quantité des données, en augmentant la taille du corpus d'apprentissage avec le corpus RAW. De la sorte nous avons pu évaluer l'effet de

l'ajout de données bruitée au modèle, et constater que cet ajout avait plutôt tendance à dégrader les performances globales.

Nous avons constaté par comparaison que l'approche symbolique obtenait de meilleurs résultats dans les deux méthodes, 96.56% de F1 moyenne.

Concernant l'approche symbolique, il y a encore des cas que nous n'avons pas les moyens de traiter, notamment pour gérer les parties correspondant à des lettres. Pour l'instant, l'approche symbolique n'est pas très efficace pour les reconnaître, et l'approche neuronal apparait plus robuste pour contourner cet écueil.

En utilisant l'approche symbolique, nous avons constaté que les règles que nous avions élaborées ne couvraient pas très bien certaines situations. Nous devions affiner certaines des règles que nous avions déjà. Nous avons découvert que cela est dû au petit nombre de romans dans notre corpus de développement, et nous pensons qu'il faudrait augmenter le nombre de romans dans le corpus de développement afin de mieux développer des règles pour couvrir plus de cas de figure, et inclure plus d'occurrences de discours direct.

Pour l'apprentissage automatique, nous n'avons procédé qu'à des ajustements simples des paramètres de *epoch*, et d'autres paramètres, tels que l'efficacité de l'apprentissage, peuvent être ajustés à l'avenir pour voir si la reconnaissance automatique de discours direct par le modèle peut être rendue plus efficace. Enfin, il serait intéressant de corriger manuellement une partie du corpus RAW afin de déterminer l'évolution des performances avec un corpus d'entrainement plus grand mais non bruité. De la sorte on pourrait obtenir de meilleurs résultats que le modèle actuel.

Bibliographie

Abeillé, A., & Godard, D. (2021). *La grande grammaire du français*. Éditions Actes Sud, 1938-1939, 2053-2071.

Alrahabi, M., Desclés, J. P., & Suh, J. (2010). Direct Reported Speech in multilingual texts: Automatic annotation and semantic categorization. *Twenty-Third International FLAIRS Conference*.

Aurey, X. (2021). *Créer une Expression régulière (REGEX) de détection des jurisprudences dans un texte*. Fondamentaux. https://www.fondamentaux.org/2021/creer-une-expression-reguliere-regex-de-detection-des-jurisprudences-dans-un-texte/

Authier-Revuz, J. (1992). Repères dans le champ du discours rapporté. *L'information grammaticale*, 55(1), 38-42.

Barnas, M. (2017). *Rapport de stage*, Université Grenoble Alpes.

Bessac, A. (1999). Laurence Rosier (Ed.) Le discours rapporté. Histoire, théories, pratiques. *Cahiers de praxématique*, (32), 228-231.

Brunner, A., Tu, N. D. T., Weimer, L., & Jannidis, F. (2020). To BERT or not to BERT-Comparing Contextual Embeddings in a Deep Learning Architecture for the Automatic Recognition of four Types of Speech, Thought and Writing Representation, *SwissText/KONVENS*.

Byszuk, J., Woźniak, M., Kestemont, M., Leśniak, A., Łukasik, W., Šeļa, A., & Eder, M. (2020). Detecting direct speech in multilingual collection of 19th-century novels. *Proceedings of 1st Workshop on Language Technologies for Historical and Ancient Languages (LT4HALA), Marseille, France. European Language Resources Association (ELRA)*, 100-104.

Ceccaldi-Hamet, A., & Lacaze, G. (2020). Le discours rapporté et l'expression de la subjectivité. *E-rea. Revue électronique d'études sur le monde anglophone*, 17.2.

de Mattia-Viviès, M. (2003). Discours indirect libre et effet de discours indirect libre. Essai de formalisation énonciative. *Numéro spécial du Bulletin de la Société de Stylistique Anglaise, Atelier Intégré de Reprographie de l'université de Paris X-Nanterre et SSA*, 107-142

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diwersy, S., Gonon, L., Goossens, V., Kraif, O., Novakova, I., Sorba, J., & Vidotto, I. (2021). *La phraséologie du roman contemporain dans les corpus et les applications de la PhraseoBase. Corpus*. OpenEdition Journals. http://journals.openedition.org/corpus/6101.

Dolamic, L., & Augustyn, M. (2017). Repérage semi-automatique du discours direct : acquisition et évaluation sur corpus Sermo (XVI-XVIIIe siècles). *Actes des 9èmes Journées Internationales de la Linguistique de corpus*, 35.

Erenup (2021), *Natural Language Processing with transformers*. Zhihu. https://zhuanlan.zhihu.com/p/367423731.

Giguet, E., & Lucas, N. (2004). La détection automatique des citations et des locuteurs dans les textes informatifs. *Le discours rapporté dans tous ses états : Question de frontières*, 410-418.

Grossmann, F. (1999). Rosier L. (1999) : Le Discours Rapporté, Histoire, théories, pratiques. *Repères. Recherches en didactique du français langue maternelle*, 19(1), 234-238.

He, H., Barbosa, D., & Kondrak, G. (2013, August). Identification of speakers in novels. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1312-1320.

HUGGING FACE (2022), *Datasets*. https://huggingface.co/docs/datasets/index.

HUGGING FACE (2022), *Transformers*. https://huggingface.co/docs/transformers/index

Jannidis, F., Zehe, A., Konle, L., Hotho, A., & Krug, M. (2018). Analysing direct speech in German Novels. *Konferenzabstracts der DHd 2018. Kritik der Digitalen Vernunft*.

Jia, H., & Luo, Z. (2019). A Study on Quotation Recognition Based on Sequence Labeling, *Journal of Chinese Information Processing*.

July, J. (2016). Le discours direct libre entre imitation naturelle de l'oral et ambiguïsation narrative. *Genres littéraires et pratiques énonciatives*, 117-130.

Krestel, R., Bergler, S., & Witte, R. (2008). Minding the source: Automatic tagging of reported speech in newspaper articles. *Reporter, 1(5)*, 4.

Kurfali, M., & Wirén M. (2020). Zero-shot cross-lingual identification of direct speech using distant supervision. *Proceedings of LaTeCH-CLfL 2020.*

Lacaze, G. (2011). De l'incise au segment contextualisant : un changement d'horizon dans l'introduction du discours direct. *Etudes de stylistique anglaise*.

Le Pesant, D. (2012). Sur les introducteurs de discours rapporté au style direct. *HAL*.

LI Li (2021), *Huggingface Transformer*. http://fancyerii.github.io/2021/05/11/huggingface-transformers-1/.

Liang, J., Dhillon, N., & Koperski, K. (2010). A large-scale system for annotating and querying quotations in news feeds. *Proceedings of the 3rd International Semantic Search Workshop*, 1-5.

Martin, L., Muller, B., Suárez, P.J.O., Dupont, Y., Romary, L., Clergerie, É.V., Seddah,D., & Sagot, B. (2020). CamemBERT: a Tasty French Language Model. *arXiv preprint arXiv:1911. 03894.*

Mourad, G., & Desclés, J. P. (2004). Identification et extraction automatique des informations citationnelles dans un texte. *López-Muñoz et al*, 397-409.

Oberle, B. (2019). Détection automatique de chaînes de coréférence pour le français écrit. *Conférence sur le Traitement Automatique des Langues Naturelles (TALN-RECITAL) 2019*.

Pareti, S., O'Keefe, T., Konstas, I., Curran, J.-R., & Koprinska, I. (2013). Automatically detecting and attributing indirect quotations. *Proceedings of Empirical Methods on Natural Language Processing (EMNLP).*

Poulard, F., Waszak, T., Hernandez, N., & Bellot, P. (2008). Repérage de citations, classification des styles de discours rapporté et identification des constituants citationnels en écrits journalistiques. *Traitement Automatique des Langues Naturelles, Jun 2008, Avignon, France.*

Pouliquen, B., Steinberger, R., & Best, C. (2007). Automatic Detection of Quotations in Multilingual News. *Proceedings of the International Conference Recent Advances in Natural Language Processing.*

Prak-Derrington, E. (2004). La fausse simplicité du discours direct. Propriétés de la parole alternée dans le dialogue romanesque. *Cahiers d'études germaniques*, (47-2), 19-32.

Rabatel, A. (2021). Discours direct libre et parole intérieure. *Pratiques. Linguistique, littérature, didactique*, 191-192.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language Models are Unsupervised Multitask Learners. *OpenAI blog*, *1*(8), 9.

Ratinov, L., & Roth, D. (2009). Design challenges and misconceptions in named entity recognition. *Proceedings of the thirteenth conference on computational natural language learning (CoNLL-2009)*, 147-155.

Reggiani, C. (2010). Les discours directs libres dans la prose narrative de Stendhal. *2ème Congrès Mondial de Linguistique Française*, 080.

Riegel, M., Pellat, J. C., & Rioul, R. (1994). *Grammaire méthodique du français*. Collection Linguistique Nouvelle. Presses Universitaires de France, Paris,1009-1014.

Schöch, C., Schlör, D., Popp, S., Brunner, A., Henny, U., & Tello, J. C. (2016). Straight Talk! Automatic Recognition of Direct Speech in Nineteenth-Century French Novels. *Digital Humanities 2016*, 346-353.

Sini, A., Delais-Roussarie, E., & Lolive, D. (2018). Annotation automatique des types de discours dans des livres audio en vue d'une oralisation par un système de synthèse. *TALN-RECITAL 2018, Rennes, France*.

Tu, N.-D.-T., Krug, M., & Brunner, A. (2019). Automatic recognition of direct speech without quotation marks. A rule-based approach. *Abstract zur Konferenz Digital Humanities im deutschsprachigen Raum 2019.*

Tunstall, L., von Werra, L., & Wolf, T. (2022). *Natural language processing with transformers*. " O'Reilly Media, Inc.".

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv preprint arXiv:1702.01923*.

Table des illustrations

Table des annexes

Annexe 1
Le script consistant à projeter l'attribut “dd” de l'élément <*p*> vers l'élément <*s*>

Algorithm - Projeter l'attribut “dd” de l'élément <p> vers l'élément <s>

Input:
Fichier d’entrée (fichier de roman, format XML)
Output :
Fichier de sortie (format XML, L'attribut de type de toutes les balises <s> qui répondent aux exigences se voit attribuer la valeur type= "dd")

```
1  Import module xml.dom.minidom et stanza

   # Utiliser module xml.dom.minidom pour parser le fichiers de Input
2  DOMTree ← xml.dom.minidom.parse(Fichier d’entrée)
3  collection ← DOMTree.documentElement

   # Obtenir toutes les informations de la balise <p>
4  PP ← collection.getElementsByTagName("p")

   # Parcourir sur les balises <p> pour obtenir des valeurs
5  Pour chaque p de PP :
       #Obtenir la valeur de l’attribut "type" de la balise <p>
6  1   type ← p.getAttribute("type")
       # Déterminer si l'attribut "type" de la balise <p> est "dd", si c'est le cas, ajouter l'attribut
       "type" à toutes les balises <s> contenues dans la balise <p> et attribuez la valeur "dd".
7  2   Si type == "dd" Alors :
8  3   1   SS ← p.getElementsByTagName("s")
9  4   2   Pour chaque s de SS :
10 5   3   1   s.setAttribute("type","dd")
11 6   Fin Si

   # Créer des fichiers de sortie
12 Fichier de sortie ← écrire(le nom de sortie, "w")
```

Annexe 2
Le script du traitement ultérieur du corpus initial

Algorithm- Le script du traitement ultérieur du corpus initial_avec l'attribute *type*

Input :
Fichier d'entrée (fichier de roman, format XML)
Output :
Fichier de sortie (format XML, au niveaux token, avec l'attribut "*type*")

```
Import module xml.dom.minidom et stanza
# Télécharger le français de stanza
stanza.dowload ('fr')
# On va utiliser les fonctions 'tokenize','lemma','pos' et 'depparse' de stanza pour analyser le roman
nlp ← stanza.Pipeline('fr', processors="tokenize,lemma,pos,depparse")

# Utiliser module xml.dom.minidom pour parser le fichiers de Input
DOMTree ← xml.dom.minidom.parse(Fichier d'entrée)
collection ← DOMTree.documentElement

# Obtenir toutes les informations de la balise <p>
PP ← collection.getElementsByTagName("p")

# Initialiser la valeur de l'id dans la balise <t>, en s'assurant qu'elle commence à 1
t_id ← 1

# Initialisation du format du document de sortie
domTree ← Document()
rootNode ← domTree.createElement("div")
domTree.appendChild(rootNode)

# Parcourir sur les balises <p> pour obtenir des valeurs
    Pour chaque p de PP :
    # Obtenir la valeur de l'attribut "id" de la balise <p>
        id_p ← p.getAttribute("id")
    # Créer des balises <p> pour le document de sortie et attribuer des valeurs
        p_node ← domTree.createElement("p")
        p_node.setAttribute("id",id_p)
        rootNode.appendChild(p_node)
    # Obtenir les balises <s> contenus dans chaque balise <p> dans le document d'entrée
        SS ← p.getElementsByTagName("s")
        Pour chaque c de SS :
        # Obtenir la valeur de l'attribut "type" et "id" de la balise <s>
            type ← c.getAttribute("type")
            id_s ← c.getAttribute("id")
        # Créer des balises <s> pour le document de sortie et attribuer des valeurs
            s_node ← domTree.createElement("s")
            s_node.setAttribute("id",id_s)
            p_node.appendChild(s_node)
```

```
                    # Obtenir la valeur de chaque balise <s> du document d'entrée (contenu de la phrase)
23  12  6           contenu ← c.firstChild.data
                    # Utiliser Stanza pour analyser les phrases
24  13  7           doc ← nlp(contenu)
25  14  8           Pour chaque sentence de doc.sentences :
26  15  9  1          doc ← nlp(contenu)
27  16  10 2          Pour chaque word de sentence.words :
                      # Créer des balises <t> pour le document de sortie et attribuer des valeurs
28  17  11 3  1         t_node ← domTree.createElement("t")
29  18  12 4  2         t_node.setAttribute("id","t"+str(t_id))
                        # L'attribuer "num" et la valeur
30  19  13 5  3         t_node.setAttribute("num",str(word.id))
                        # L'attribuer "l" (lemma) et la valeur
31  20  14 6  4         t_node.setAttribute("l",word.lemma)
                        # L'attribuer "c" (catégorie) et la valeur
32  21  15 7  5         t_node.setAttribute("c",word.pos)
                        # L'attribuer "f" (format, les caractéristiques morphologiques du token) et la valeur
33  22  16 8  6         t_node.setAttribute("f",word.feats)
                        # L'attribuer "type" et la valeur
34  23  17 9  7         t_node.setAttribute("type",type)
                        # Contenu de la balise <t>
35  24  18 10 8         t_text ← domTree.createTextNode(word.text)
                        # Numérotation incrémentielle des balises <t>
36  25  19 11 9         t_id ← t_id+1
                        # Ajouter des balises <t>
37  26  20 12 10        s_node.appendChild(t_node)
38  27  21 13 11        t_node.appendChild(t_text)

# Créer des fichiers de sortie
39  Fichier de sortie ← écrire(le nom de sortie, "w")
```

Algorithm - Le script du traitement ultérieur du corpus initial_sans l'attribute *type*

Input:
Fichier d'entrée (fichier de roman, format XML)
Output :
Fichier de sortie (format XML, au niveaux token, sans l'attribut "*type*")

```
1  Import module xml.dom.minidom et stanza
# Télécharger le français de stanza
2  stanza.dowload ('fr')
# On va utiliser les fonctions 'tokenize','lemma','pos' et 'depparse' de stanza pour analyser le roman
3  nlp ← stanza.Pipeline('fr', processors="tokenize,lemma,pos,depparse")

# Utiliser module xml.dom.minidom pour parser le fichiers de Input
4  DOMTree ← xml.dom.minidom.parse(Fichier d'entrée)
5  collection ← DOMTree.documentElement

# Obtenir toutes les informations de la balise <p>
6  PP ← collection.getElementsByTagName("p")

# Initialiser la valeur de l'id dans la balise <t>, en s'assurant qu'elle commence à 1
```

```
t_id ← 1

# Initialisation du format du document de sortie
domTree ← Document()
rootNode ← domTree.createElement("div")
domTree.appendChild(rootNode)

# Parcourir sur les balises <p> pour obtenir des valeurs
Pour chaque p de PP :
    # Obtenir la valeur de l'attribut "id" de la balise <p>
    id_p ← p.getAttribute("id")
    # Créer des balises <p> pour le document de sortie et attribuer des valeurs
    p_node ← domTree.createElement("p")
    p_node.setAttribute("id",id_p)
    rootNode.appendChild(p_node)
    # Obtenir les balises <s> contenus dans chaque balise <p> dans le document d'entrée
    SS ← p.getElementsByTagName("s")
    Pour chaque c de SS :
        # Obtenir la valeur de l'attribut "id" de la balise <s>
        id_s ← c.getAttribute("id")
        # Créer des balises <s> pour le document de sortie et attribuer des valeurs
        s_node ← domTree.createElement("s")
        s_node.setAttribute("id",id_s)
        p_node.appendChild(s_node)
        # Obtenir la valeur de chaque balise <s> du document d'entrée (contenu de la phrase)
        contenu ← c.firstChild.data
        # Utiliser Stanza pour analyser les phrases
        doc ← nlp(contenu)
        Pour chaque sentence de doc.sentences :
            doc ← nlp(contenu)
            Pour chaque word de sentence.words :
                # Créer des balises <t> pour le document de sortie et attribuer des valeurs
                t_node ← domTree.createElement("t")
                t_node.setAttribute("id","t"+str(t_id))
                # L'attribuer "num" et la valeur
                t_node.setAttribute("num",str(word.id))
                # L'attribuer "l" (lemma) et la valeur
                t_node.setAttribute("l",word.lemma)
                # L'attribuer "c" (catégorie) et la valeur
                t_node.setAttribute("c",word.pos)
                # L'attribuer "f" (format, les caractéristiques morphologiques du token) et la valeur
                t_node.setAttribute("f",word.feats)
                # Contenu de la balise <t>
                t_text ← domTree.createTextNode(word.text)
                # Numérotation incrémentielle des balises <t>
                t_id ← t_id+1
                # Ajouter des balises <t>
                s_node.appendChild(t_node)
                t_node.appendChild(t_text)
```

```
# Créer des fichiers de sortie
Fichier de sortie ← écrire(le nom de sortie, "w")
```

Annexe 3
La liste complète pour *VerbDic*

La liste complète pour *VerbDic*	
Verbes non-pronominaux	**Verbes pronominaux**
aboyer	abstenir
accorder	agacer
acquiescer	angoisser
admettre	assurer
affecter	chapitrer
affirmer	dire
ajouter	écrier
annoncer	égayer
apostropher	emporter
apprécier	empresser
approuver	enquérir
appuyer	enthousiasmer
arguer	esclaffer
cracher	esclaffer
assurer	étonner
attaquer	étrangla
bafouiller	exciter
balbutier	exclamer
bêler	excuser
beugler	exhorter
blêmir	impatienter
bougonner	indigner
brailler	informer
bredouiller	inquiéter
charrier	insurgea
chuchoter	interposer
clamer	irriter
commander	lamenter
commenter	moquer
conclure	morigéner
confier	opposer
conjurer	présenter
constater	réjouir
continuer	rembrunir
corriger	répéter
couper	stimuler
crachoter	apprendre
crier	avouer
critiquer	faire
déclarer	entêter
décourager	
décréter	

demander	
démarrer	
désigner	
devoir	
dire	
enjoindre	
entendre	
entreprendre	
éructer	
estimer	
expliquer	
exploser	
exposer	
exulter	
finir	
formuler	
fredonner	
fulminer	
geindre	
gémir	
glapir	
glisser	
glousser	
gouailler	
grimacer	
grincer	
grogner	
grommeler	
gronder	
gueuler	
haleter	
hasarder	
hurler	
implorer	
indiquer	
insister	
interpeller	
interroger	
intervenir	
ironiser	
jeter	
jubiler	
lâcher	
lancer	
marmonner	
maugréer	
mentir	
minauder	
murmurer	
narguer	
noter	
objecter	
observer	

opiner	
ordonner	
oser	
pâlir	
penser	
persifler	
pester	
piaffer	
plaider	
plaisanter	
pleurnicher	
postillonner	
pouffer	
poursuivre	
préciser	
prédire	
presser	
prétexter	
prier	
proférer	
prononcer	
proposer	
protester	
questionner	
railler	
râler	
rassurer	
recommencer	
reconnaître	
relancer	
remarquer	
renchérir	
repartir	
répéter	
répliquer	
répondre	
reprendre	
résoudre	
rétorquer	
ricaner	
rigoler	
riposter	
risquer	
roucouler	
rugir	
saluer	
sangloter	
secouer	
siffler	
soliloquer	
songer	
souffler	
soupirer	

stopper	
suggérer	
supplier	
susurrer	
taquiner	
tempêter	
traduire	
trancher	
triompher	
trouver	
hoqueter	
minalder	
vociférer	

Annexe 4
Le script d'identification automatique du discours direct (l'approche symbolique)

Algorithm - Règle 1

```
Input :
   Fichier d'entrée (fichier de roman, format XML)
Output :
  step1.1_out.xml (format XML, fichier de sortie)
Import module xml.dom.minidom et re
# Créer la fonction pour mettre en œuvre la reconnaissance de la Règle 1
regex ← r"—"
fonction identifier (content) {
    matches ← re.finditer(regex,content,re.MULTILINE)
    Pour chaque i de marches :
        Si i.group() == "—" Alors
            retourne True
        Sinon
            retourne False
        Fin Si
}
# Utiliser module xml.dom.minidom pour parser le fichiers de Input
DOMTree ← xml.dom.minidom.parse(Fichier d'entrée)
collection ← DOMTree.documentElement

# Obtenir toutes les informations de la balise <p>
PP ← collection.getElementsByTagName("p")
# Parcourir sur les balises <p>
Pour chaque p de PP :
    # Parcourir les balises <t> sous chaque balise <p>
    TT ← p.getElementsByTagName("t")
    # Parcourir les balises <t> pour obtenir les valeurs
    Pour chaque t de TT :
        contenu ← t.childNodes[0].data
        # Déterminer si le token satisfait à la Règle 1.
        Si identifier (contenu) Alors
            Pour chaque j de TT :
                # Si c'est le cas, ajouter un attribut "type" au token et lui attribue la valeur "dd"
                ADD ← DOMTree.createAttribute("type")
                j.setAttributeNode(ADD)
                j.setAttribute("type","dd")
        Sinon
            break
        Fin Si

# Créer des fichiers de sortie
```

```
19 step1.1_out.xml ← écrire(fichier de sortie, "w")
```

Algorithm - Règle 2

Input:
step1.1_out.xml (fichier d'entrée, format XML)
Output :
step1_out.xml (format XML, fichier de sortie)

```
1  Import module xml.dom.minidom et re
# Créer la fonction pour mettre en œuvre la reconnaissance de la Règle 2
2  regex ← r"«|»"
# Pour trouver des «...», si «...» sont précédées de " ] ", alors le contenu de «...» ne fait pas de
discours direct. Ex : [17] « Nous sommes françaises », en suédois.
3  regex_star ←r"\]"
4  fonction identifier (content, reg) {
       matches ← re.finditer(reg,content,re.MULTILINE)
       Pour chaque i de marches :
        # Trouvez les tokens entre les «...»
          Si i.group() != None Alors
             retourne True
          Sinon
             retourne False
          Fin Si
}
# Utiliser module xml.dom.minidom pour parser le fichiers de Input
5  DOMTree ← xml.dom.minidom.parse(step1.1_out.xml)
6  collection ← DOMTree.documentElement
# Variable pour déterminer si "«" ou "»" est trouvée
7  head ← False

# Obtenir toutes les informations de la balise <p>
8  PP ← collection.getElementsByTagName("p")
# Parcourir sur les balises <p>
9  | Pour chaque p de PP :
   | # Parcourir les balises <t> sous chaque balise <p>
10 | 1 | head ← False
11 | 2 | TT ← p.getElementsByTagName("t")
   |   | # Parcourir les balises <t> pour obtenir le contenu
12 | 3 | Pour chaque t de TT :
13 | 4 | 1 | contenu ← t.childNodes[0].data
   |   |   | # Déterminer si le token satisfait à la Règle 2 (trouver "«" et "»")
14 | 5 | 2 | ide ← identifier (contenu, regex)
15 | 6 | 3 | Si ide Alors
16 | 7 | 4 | 1 | Si not head Alors  # Trouver le "«"
17 | 8 | 5 | 2 | 1 | head ← True
18 | 9 | 6 | 3 | Sinon  #Trouver le "»"
19 | 10 | 7 | 4 | 2 | head ← False
   |    |   |   |   | # Ajouter un attribut "type" au token et lui attribue la valeur "dd"
20 | 11 | 8 | 5 | 3 | ADD ← DOMTree.createAttribute("type")
21 | 12 | 9 | 6 | 4 | t.setAttributeNode(ADD)
22 | 13 | 10 | 7 | 5 | t.setAttribute("type","dd")
```

```
                    Fin Si
                Sinon
            Si head Alors # C'est pas le "«" et le "»", ce sont les toknes entre «...»
                # Ajouter un attribut "type" au token et lui attribue la valeur "dd"
                ADD ← DOMTree.createAttribute("type")
                t.setAttributeNode(ADD)
                t.setAttribute("type","dd")
            Fin Si
        Fin Si

# Rechercher une deuxième fois les «...» liées à "]" et changer la valeur d'attribut "type" du token en "/". Parce que ces tokens ne sont pas de discours direct

# Parcourir sur les balises <p>
Pour chaque p de PP :
    # Parcourir les balises <t> sous chaque balise <p>
    TT ← p.getElementsByTagName("t")
    situation ← False
    first1 ← False
    head1 ← False
    # Parcourir les balises <t> pour obtenir le contenu
    Pour chaque t de TT :
        contenu ← t.childNodes[0].data
        # Déterminer s'il y a "]"
        ide_star ← identifier(contenu,regex_star)
        Si ide_star Alors
            situation ← True
            first1 ← True
        Sinon
        # Trouver le "«" et le "»"
            ide1 ← identifier(contenu,regex)
            Si ide1 Alors
                Si not head1 ET contenu == "«" Alors
                    head1 ← True
                Sinon
                    head1 ← False  # Trouver le "»"
                    Si first1 Alors
                        situation ← False
                    Fin Si
                Fin Si
                Si situation Alors
                # changer la valeur d'attribut "type" du token en "/"
                    t.setAttribute("type","/")
                Fin Si
                Si not head1 Alors
                    situation ← False
                Fin Si
```

```
                Sinon  # Trouver pas le "«" et le "»"
                    Si first1 Alors
                        situation ← False
                        first1 ← False
                    Sinon
                        Si head1 ET situation Alors. # Les tokens entre "«… »"
                        # changer la valeur d'attribut "type" du token en "/"
                            t.setAttribute("type","/")
                    Fin Sin
                Fin Sin
            Fin Sin

# Créer des fichiers de sortie
step1_out.xml ← écrire(fichier de sortie, "w")
```

Algorithm - Règle 3

```
Input:
   step1_out.xml (fichier d'entrée, format XML)
Output :
  Step2_out.xml (format XML, fichier de sortie)
Import module xml.dom.minidom et re
# Créer la fonction pour mettre en œuvre la reconnaissance de la Règle 3
regex_1 ← r" \( | \) "
regex_2 ← r" \[ | \] "
fonction identifier (content, regex) {
    matches ← re.finditer(regex,content,re.MULTILINE)
    Pour chaque i de marches :
     # Trouvez les tokens entre les (...) et […]
        Si i.group() != None Alors
            retourne True
        Sinon
            retourne False
        Fin Si
}
# Utiliser module xml.dom.minidom pour parser le fichiers de Input
DOMTree ← xml.dom.minidom.parse(step1_out.xml)
collection ← DOMTree.documentElement
# Variable pour déterminer si "[", " (" ou "]", " )" est trouvée
head ← False

# Obtenir toutes les informations de la balise <p>
PP ← collection.getElementsByTagName("p")
# Parcourir sur les balises <p>
    Pour chaque p de PP :
    # Parcourir les balises <t> sous chaque balise <p>
        TT ← p.getElementsByTagName("t")
        # Parcourir les balises <t> pour obtenir le contenu et les valeurs
        Pour chaque t de TT :
            type ← t.getAttribute("type")
```

```
                    id ← t.getAttribute("id")
                    # Déterminer si l'attribut "type" est marqué comme "dd"
                    Si type == "dd" Alors
                        contenu ← t.childNodes[0].data
                        ide ← identifier(contenu,regex_2) OU identifier(contenu,regex_1)
                        Si ide Alors
                            Si not head Alors
                                head ← True  # Trouver "[" et " ("
                            Sinon
                                head ← False  # Trouver "]" et " )"
                            Fin Si
                            t.setAttribute("type","/")
                        Sinon
                            Si head Alors  #Trouver les tokens entre "[...]" ou "(...)"
                                t.setAttribute("type","/")
                            Fin Si
                        Fin Si
                    Fin Si

# Créer des fichiers de sortie
step2_out.xml ← écrire(fichier de sortie, "w")
```

Algorithm - Règle 4 Pron

Input:
step2_out.xml (fichier d'entrée, format XML)
Output :
step3.1_out.xml (format XML, fichier de sortie)

```
Import module xml.dom.minidom et re
# Créer la fonction pour mettre en œuvre la reconnaissance de la Règle 4
regex_star ← r",|!|\?"
regex_end ← r",|\.|!|\?"
fonction identifier (content, regex) {
    matches ← re.finditer(regex,content,re.MULTILINE)
    Pour chaque i de marches :
        Si i.group() != None Alors
            retourne True
        Sinon
            retourne False
        Fin Si
}
# Liste VerbDic
Verb_List ←
    ["aboyer","abstenir","accorder","agacer","acquiescer","angoisser","admettre","assurer","affecter","chapit
    rer","affirmer","dire","ajouter","écrier","annoncer","égayer","apostropher","emporter","apprécier","empr
    esser","approuver","enquérir","appuyer","enthousiasmer","arguer","esclaffer","cracher","esclaffer","assu
    rer","étonner","attaquer","étrangla","bafouiller","exciter","balbutier","exclamer","bêler","excuser","beug
    ler","exhorter","blêmir","impatienter","bougonner","indigner","brailler","informer","bredouiller","inquié
    ter","charrier","insurgea","chuchoter","interposer","clamer","irriter","commander","lamenter","comment
    er","moquer","conclure","morigéner","confier","opposer","conjurer","présenter","constater","réjouir","c
    ontinuer","rembrunir","corriger","répéter","couper","stimuler","crachoter","crier","critiquer","déclarer","
    décourager","décréter","demander","démarrer","désigner","devoir","dire","enjoindre","entendre" ,"entre
    prendre","éructer","estimer","expliquer","exploser","exposer","exulter","finir","formuler","fredonner","f
```

```
ulminer","geindre","gémir","glapir","glisser","glousser","gouailler","grimacer","grincer","grogner","grommeler","gronder","gueuler","haleter","hasarder","hurler","implorer","indiquer","insister","interpeller","interroger","intervenir","ironiser","jeter","jubiler","lâcher","lancer","marmonner","maugréer","mentir","minauder","murmurer","narguer","noter","objecter","observer","opiner","ordonner","oser","pâlir","penser","persifler","pester","piaffer","plaider","plaisanter","pleurnicher","postillonner","pouffer","poursuivre","préciser","prédire","presser","prétexter","prier","proférer","prononcer","proposer","protester","questionner","railler","râler","rassurer","recommencer","reconnaître","relancer","remarquer","renchérir","repartir","répéter","répliquer","répondre","reprendre","résoudre","rétorquer","ricaner","riposter","risquer","rougir","saluer","sangloter","secouer","siffler","soliloquer","songer","souffler","soupirer","stopper","suggérer","supplier","susurrer","taquiner","tempêter","traduire","trancher","triompher","trouver","vociférer","apprendre","avouer","entêter","rigoler","hoqueter","faire","minalder"]

# Utiliser module xml.dom.minidom pour parser le fichiers de Input
DOMTree ← xml.dom.minidom.parse(step2_out.xml)
collection ← DOMTree.documentElement

# Créer la fonction pour mettre en œuvre la reconnaissance de la situation Pron
fonction PRON_VERB (PP) {
# L'ensemble "num" du token où se trouve Pron
  ID_ALL ← []
  Pour chaque p de PP :
    head ← False
    situation ← False
    pron ← False
    end ← False
# Parcourir les balises <t> sous chaque balise <p>
    TT ← p.getElementsByTagName("t")
    Pour chaque i de TT :
# Obtennir "type","catégorie","lemma" et "forma" du token pour faciliter une détermination ultérieure
      type ← i.getAttribute("type")
      category ← i.getAttribute("c")
      lemma ← i.getAttribute("l")
      forma ← str(i.getAttribute("f"))
# déterminer si le "type" est marqué comme "dd", est "dd" seulement juge, gagner du temps.
      Si type == "dd" Alors
        cc ← i.childNodes[0].data
        Si not head Alors
          ide ← identifier(regex_start,cc)  # Trouver le signes de début
        Sinon
          ide ← identifier(regex_end,cc)  # Trouver le signes de fin
        Fin Sin
        Si ide Alors  # Si on le trouve, changer le valeur des variables
          situation ← False
          head ← False
          pron ← False
          end ← False
          Si not head Alors
            head ← True
          Fin Si
        Sinon
          Si not end Alors
            Si head Alors  # Conforme au token après le signe de ponctuation de la Règle 4
              Si not situation Alors
```

```
                Si not pron Alors
            # Si le token est pron, on porte un jugement. Cela ne peut pas être « tu / vous / on»
                    Si category == "PRON" ET cc !=
                    "tu/TU/Vous/vous/On/on"
                       pron ← True
                    Sinon
                       end ← True
                    Fin Si
                Sinon
            # Voir si un token après PRON est un verbe
                    Si category == "VERB" Alors
            # Regarder si le lemma du verbe est dans la liste Verb_List
                       Si lemma dans Verb_List Alors
            # Vérifier si la conjugaison du verbe correspond aux exigences
                          Si "Mood=Ind" et "Person=3 ou Person=1" et
                          "Tense=Past ou Tense=Pres ou Tense = Imp" dans
                          forma Alors
                             situation ← True
                             i.setAttribute("type","/")
                             id ←i.getAttribute("id")
                             ID_ALL.ajouter(id)
                          Fin Si
                       Sinon
                          end ← True
                       Fin Si
                    Sinon
                       end ← True
                    Fin Si
                 Fin Si
              Sinon
                 i.setAttribute("type","/")
              Fin Si
           Fin Si
        Sinon
           Si situation Alors
              i.setAttribute("type","/")
           Fin Si
        Fin Si
     Fin Sin
  Fin Sin
  retour ID_ALL

# Obtenir toutes les informations de la balise <p>
PP ← collection.getElementsByTagName("p")
# Retourner les "id" des tokens qui correspondent aux exigences, puis parcourir pour trouver ces
tokens et changer la valeur de leur attribut "type" de "dd" en "/"
pr_id ← PRON_VERB(PP)

Pour chaque k de pr_id :
   # Variable qui indique si le token a été trouvé
   find ← False
   k ← int(k.strip("t"))-1
```

```
                String ← "t"+str(k)

                Pour chaque p de PP :
                # Parcourir toutes les balises <t>
                    TT ← p.getElementsByTagName("t")
                    Pour chaque i de TT :
                    # Obtenir la valeur de l'attribut "id" de la balise <t>
                        lid ← i.getAttribute("id")
                        # Trouver les tokens qui correspondent aux exigences
                        Si lid == String Alors
                            i.setAttribute("type","/")
                            find ← True
                            break
                        Fin Si
                    Si find Alors
                       break
                    Fin Si

# Créer des fichiers de sortie
step3.1_out.xml ← écrire(fichier de sortie, "w")
```

Algorithm - Règle 4 Verbe

Input:
step3.1_out.xml (fichier d'entrée, format XML)
Output :
step3_out.xml (format XML, fichier de sortie)

```
Import module xml.dom.minidom et re
# Créer la fonction pour mettre en œuvre la reconnaissance de la Règle 4
regex_star ← r",|!|\?"
regex_end ← r",|\.|!|\?"
fonction identifier (content, regex) {
    matches ← re.finditer(regex,content,re.MULTILINE)
    Pour chaque i de marches :
        Si i.group() != None Alors
            retourne True
        Sinon
            retourne False
        Fin Si
}
# Liste VerbDic
Verb_List ←
    ["aboyer","abstenir","accorder","agacer","acquiescer","angoisser","admettre","assurer","affecter","chapit
    rer","affirmer","dire","ajouter","écrier","annoncer","égayer","apostropher","emporter","apprécier","empr
    esser","approuver","enquérir","appuyer","enthousiasmer","arguer","esclaffer","cracher","esclaffer","assu
    rer","étonner","attaquer","étrangla","bafouiller","exciter","balbutier","exclamer","bêler","excuser","beug
    ler","exhorter","blêmir","impatienter","bougonner","indigner","brailler","informer","bredouiller","inquié
    ter","charrier","insurgea","chuchoter","interposer","clamer","irriter","commander","lamenter","comment
    er","moquer","conclure","morigéner","confier","opposer","conjurer","présenter","constater","réjouir","c
    ontinuer","rembrunir","corriger","répéter","couper","stimuler","crachoter","crier","critiquer","déclarer","
    décourager","décréter","demander","démarrer","désigner","devoir","dire","enjoindre","entendre" ,"entre
    prendre","éructer","estimer","expliquer","exploser","exposer","exulter","finir","formuler","fredonner","f
    ulminer","geindre","gémir","glapir","glisser","glousser","gouailler","grimacer","grincer","grogner","gro
    mmeler","gronder","gueuler","haleter","hasarder","hurler","implorer","indiquer","insister","interpeller","
```

```
interroger","intervenir","ironiser","jeter","jubiler","lâcher","lancer","marmonner","maugréer","mentir","minauder","murmurer","narguer","noter","objecter","observer","opiner","ordonner","oser","pâlir","penser","persifler","pester","piaffer","plaider","plaisanter","pleurnicher","postillonner","pouffer","poursuivre","préciser","prédire","presser","prétexter","prier","proférer","prononcer","proposer","protester","questionner","railler","râler","rassurer","recommencer","reconnaître","relancer","remarquer","renchérir","repartir","répéter","répliquer","répondre","reprendre","résoudre","rétorquer","ricaner","riposter","risquer","roucouler","rugir","saluer","sangloter","secouer","siffler","soliloquer","songer","souffler","soupirer","stopper","suggérer","supplier","susurrer","taquiner","tempêter","traduire","trancher","triompher","trouver","vociférer","apprendre","avouer","entêter","rigoler","hoqueter","faire","minalder"]

# Utiliser module xml.dom.minidom pour parser le fichiers de Input
DOMTree ← xml.dom.minidom.parse(step3.1_out.xml)
collection ← DOMTree.documentElement

# Créer la fonction pour mettre en œuvre la reconnaissance de la situation Pron
fonction VERB (PP) {
  Pour chaque p de PP :
    head ← False
    situation ← False
# Parcourir les balises <t> sous chaque balise <p>
    TT ← p.getElementsByTagName("t")
    Pour chaque i de TT :
# Obtennir "type","catégorie","lemma" et "forma" du token pour faciliter une détermination ultérieure
      type ← i.getAttribute("type")
      category ← i.getAttribute("c")
      lemma ← i.getAttribute("l")
      forma ← str(i.getAttribute("f"))
# déterminer si le "type" est marqué comme "dd", est "dd" seulement juge, gagner du temps.
      Si type == "dd" Alors
        cc ← i.childNodes[0].data
        Si not head Alors
          ide ← identifier(regex_start,cc)  # Trouver le signes de début
        Sinon
          ide ← identifier(regex_end,cc)  # Trouver le signes de fin
        Fin Sin
        Si ide Alors  # Si on le trouve, changer le valeur des variables
          situation ← False
          head ← False
          Si not head Alors
            # Trouver le signe de ponctuation de début
# Changez la valeur de head en vue de trouver le signe de ponctuation de fin la prochaine fois
            head ← True
          Fin Si
        Sinon
# C'est le token suivant le signe de ponctuation qui satisfait à la Règle 4
          Si head Alors
            Si not situation Alors
# Déterminer s'il s'agit d'un Verbe
              Si category == "VERB" Alors
# Regarder si le lemma du verbe est dans la liste Verb_List
                Si lemma dans Verb_List Alors
# Vérifier si la conjugaison du verbe correspond aux exigences
                  Si "Mood=Ind" et "Person=3 ou Person=1" et "Tense=Past
                  ou Tense=Pres ou Tense = Imp" dans forma Alors
```

```
                        Si not situation Alors
                           situation ← True
                        Sinon
                           situation ← False
                        Fin Si
                        i.setAttribute("type","/")
                     Fin Si

                  Fin Si
               Sinon
                  head ← False
               Fin Si
            Sinon
            # changer la valeur d'attribut "type" de "dd" en "/"
               i.setAttribute("type","/")
            Fin Si
         Fin Si
      Fin Si

# Obtenir toutes les informations de la balise <p>
9  PP ← collection.getElementsByTagName("p")

10 VERB(PP)


# Créer des fichiers de sortie
11 step3_out.xml ← écrire (VERB(PP), "w")
```

Annexe 5
Le script pour le format de conversion (vers *json*)

Algorithm - Le script pour le format de conversion (vers *json*)

Input:
step3_out.xml (fichier d'entrée, format XML)
Output :
Fichier de sortie (format json)

```
Import module xml.dom.minidom et json
# Utiliser module xml.dom.minidom pour parser le fichier
DOMTree ← xml.dom.minidom.parse(step3_out.xml)
collection ← DOMTree.documentElement

# Obtenir toutes les informations de la balise <s>
SS ← collection.getElementsByTagName("s")
Pour chaque s de SS :
    # créer le dic
    dic ←{}.fromkeys(("id","tokens","dd_tags"))
    first ← True
    # Ajouter "id"
    Si dic["id"] est None Alors
        dic["id"] ← "Nom de roman#"+s.getAttribute("id")
    Sinon
        dic["id"].ajouter("Nom de roman#"+s.getAttribute("id"))
    Fin Si

    cc ← s.getElementsByTagName("t")
    Pour chaque c de cc :
        token ←c.childNodes[0].data
        # Ajouter "token"
        Si dic["tokens"] est None Alors
            dic["tokens"] = [token]
        Sinon
            dic["tokens"].ajouter(token)
        Fin Si
        type ← c.getAttribute("type")
        # Ajouter "tag"
        Si dic["dd_tags"] est None Alors
            # Vérifier le premier token de la phrase, si c'est "dd", on fait 0, si non, on fait 2
            Si first Alors
                Si type == "dd" Alors
                    dic["dd_tags"] ← [0]
                Sinon
                    dic["dd_tags"] ← [2]
                Fin Si
                first ← False
            Sinon
                # Vérifier le token de la phrase, si c'est "dd", on fait 1, si non, on fait 2
```

```
                    Si type == "dd" Alors
32  18  10  8   1   | dic["dd_tags"] ← [1]
33  19  11  9       Sinon
34  20  12  10  2   | dic["dd_tags"] ← [2]
35  21  13  11      Fin Si
36  22  14      Fin Si
37  23      # Ce n'est pas la première ligne du fichier de sortie
            Sinon
38  24  15      Si first Alors
39  25  16  1       # Vérifier le premier token de la phrase, si c'est "dd", on fait 0, si non, on fait 2
                    Si type == "dd" Alors
40  26  17  2   1   | dic["dd_tags"].ajouter(0)
41  27  18  3       Sinon
42  28  19  4   2   | dic["dd_tags"].ajouter(2)
43  29  20  5       Fin Si
44  30  21  6       First ← False
45  31  22      Sinon
46  32  23  7       Si type == "dd" Alors
47  33  24  8   1   | dic["dd_tags"].ajouter(1)
48  34  25  9       Sinon
49  35  26  10  2   | dic["dd_tags"].ajouter(2)
50  36  27  11      Fin Si
51  37      Fin Si

# Créer des fichiers de sortie
52  Fichier de sortie ← écrire (nom de fichier, "a")
```

Annexe 6
Le script permettant d'obtenir *TP, FP et FP*

Algorithm - Le script permettant d'obtenir *TP, FP et FP*

Input :
step3_out.xml (fichier d'entrée, format XML)
Fichier du corpus standard (fichier d'entrée, format XML)
Output :
Des variables : *TP, FN, FP*

```
1  Import module xml.dom.minidom
   # Utiliser module xml.dom.minidom pour parser le fichier du corpus standard
2  DOMTree ← xml.dom.minidom.parse(Fichier du corpus standard)
3  collection ← DOMTree.documentElement

   # Utiliser module xml.dom.minidom pour parser le step3_out.xml
4  outTree ← xml.dom.minidom.parse(step3_out.xml)
5  collection_out ← outTree.documentElement

   # Créer deux listes pour enregistrer le nombre de tokens marqués comme "dd"
6  token_DD ← []
7  token_DD_out ← []

   # Obtenir toutes les informations de la balise <p>
8  PP ← collection.getElementsByTagName("p")
9  PP_out ← collection_out.getElementsByTagName("p")
10 Pour chaque p de PP :
11 1   # Parcourir les balises <t> sous chaque balise <p>
       TT ← p.getElementsByTagName("t")
12 2   Pour chaque i de TT :
13 3 1   type ← i.getAttribute("type")
14 4 2   Si type == "dd" Alors
15 5 3 1   id ← i.getAttribute("id")
16 6 4 2   token_DD.ajouter (id)
17 7 5   Fin Si

18 Pour chaque p_out de PP_out :
19 38  # Parcourir les balises <t> sous chaque balise <p>
       TT_out ← p_out.getElementsByTagName("t")
20 39  Pour chaque i_out de TT_out :
21 40 3   type_out ← i_out.getAttribute("type")
22 41 4   Si type_out == "dd" Alors
23 42 5 12   id_out ← i_out.getAttribute("id")
24 43 6 13   token_DD_out.ajouter (id_out)

25 num ← len(token_DD)
```

```
num_out ← len(token_DD_out)
print (num, num_out)
# Nombre de tokens pour trouver le bon marqueur. True Positive
TP ← set (token_DD).  intersection(set(token_DD_out))
print ("TP: "+str(len(TP)))
# Nombre de tokens pour trouver le False Negative
FN ← set(token_DD).difference(token_DD_out)
print ("FN: "+str(len(FN)))
# Nombre de tokens pour trouver le False Positive
FP ← set(token_DD_out).difference(token_DD)
print ("FP: "+str(len(FP)))
```

Table des matières

MOTS-CLÉS : TAL, discours direct, identification du discours direct, Apprentissage automatique, Transformer, BERT

RÉSUMÉ

Ce mémoire de Master porte sur l'identification automatique du discours direct dans un corpus de romans en français. La notion de discours direct a été théorisée à de nombreuses reprises. Et le domaine de la linguistique a décrit de manière adéquate le phénomène linguistique particulier et omniprésent qu'est le discours direct. Cependant, le discours direct se présente sous différentes formes dans différents genres de textes (par exemple, dans le roman, le journalisme, etc.). Aussi, un premier axe de notre travail est de résumer les formes de discours direct qui apparaissent dans le roman français.

Le deuxième axe porte sur le processus d'extraction de discours direct. Dans cette étude, nous avons utilisé deux méthodes pour réaliser la reconnaissance automatique de discours direct, l'approche symbolique et l'approche neuronale. Pour l'approche symbolique, nous extrayons du corpus des expressions sur le discours direct et développons des règles pertinentes par observation pour accomplir la reconnaissance automatique. Pour l'approche neuronale, nous fournissons des données provenant de différents types de romans dans l'espoir que l'apprentissage automatique puisse identifier automatiquement le discours direct.

KEYWORDS: NLP, direct speech, identification of direct speech, Machine Learning, Transformer, BERT

ABSTRACT

This Master's thesis deals with the automatic identification of direct speech in a corpus of French novels. The notion of direct speech has been theorized many times. And the field of linguistics has adequately described the particular and ubiquitous linguistic phenomenon that is direct speech. However, direct speech occurs in different forms in different genres of texts (e.g., in the novel, journalism, etc.). Thus, a first axis of our work is to summarize the forms of direct speech that appear in the French novel.

The second axis deals with the process of direct speech extraction. In this study, we used two methods to perform automatic recognition of direct speech, the symbolic approach and the neural approach. For the symbolic approach, we extract direct speech expressions from the corpus and develop relevant rules by observation to accomplish automatic recognition. For the neural approach, we provide data from different types of novels in the hope that machine learning can automatically identify direct speech.