Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Détecter les citations bibliques dans la *Mekhilta de Rabbi Yishmael* : un cas d'étude pour la supervision faible

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Recherche et Développement

par

Nicolas BONTEMPS

Directeur de mémoire :

Pierre Magistry

Encadrant :

Daniel Stoekl Ben Ezra

Année universitaire 2022/2023

Table des matières

Remerciements

Ce travail n'aurait pu être réalisé sans le soutien de Daniel Stökl Ben Ezra, responsable du projet ERC SYNERGY MIDRASH. Son expertise concernant la littérature et les manuscrits rabbiniques ainsi que sa constante disponibilité ont été d'une aide précieuse. Je le remercie de m'avoir ouvert la porte de ce nouveau monde. J'y ai rencontré de nombreux passionnés de linguistique hébraïque et d'informatique qui m'ont enrichi de leurs savoirs : Pawel Jablonski et Luigi Bambaci de l'EPHE, Avi Shmidman du projet DICTA notamment.

Je remercie également toute l'équipe du master TAL, à l'Inalco, à Nanterre et à Sorbonne Nouvelle. En premier lieu Iris Eshkol-Taravella et Mathieu Valette qui ont cru en mon projet, mais surtout Pierre Magistry, directeur de ce mémoire, pour son accompagnement et la transmission de son expérience dans le domaine du TAL. Je remercie également tous les enseignants du master, dont les cours ou les conseils ont contribué à me donner l'envie de continuer dans cette voie : Pascal Amsili, Loïc Gobol, Cyril Grouin, Damien Nouvel, Delphine Battistelli, pour ne citer qu'eux. Je n'oublie pas les étudiants rencontrés sur mon parcours, compagnons de joies et d'infortunes.

Mes remerciements s'adressent aussi à Aude Julien-Da Cruz Lima et Baptiste Buob, sans qui cela n'aurait pas été possible.

Ces remerciements seraient incomplets sans une adresse vibrante à l'association Parler en Paix qui m'a permis d'apprendre l'hébreu et l'arabe, mais surtout donné l'envie d'avancer dans la connaissance des langues sémitiques. Une mention spéciale pour Shahar Fineberg, qui a été mon professeur d'hébreu toutes ces années et a réussi à stimuler ma curiosité pour la linguistique hébraïque, ainsi qu'à Jonas Sibony, qui m'a montré la voie des liens entre arabe et hébreu, et Nadia Makouar, professeur d'arabe et première taliste sémitisante rencontrée sur mon parcours. Je pense aussi à Nurit Levy et à l'équipe du département d'hébreu de l'Université de Lille, grâce à qui j'ai pu aller plus loin dans cette étude de l'hébreu.

Enfin, je m'adresse à tous les proches, amis et familles qui m'ont accompagné, soutenus, voir même supportés dans cette aventure : Marine, Maha, Rebecca, Chloé, Grégoire, Michal et tous les autres. Merci.

Résumé

La détection des citations bibliques utilisées dans les commentaires rabbiniques médiévaux constitue un défi, tant par les caractéristiques de ces textes que par la difficulté d'accès à des corpus annotés pour l'apprentissage automatique. L'ambition de ce travail est d'évaluer la capacité de la supervision faible à générer efficacement un tel type de données. Pour cela, nous étudierons un *midrash* daté des environs des IIe-IIIe siècle, la *Mekhilta de Rabbi Yishmael*. L'exploration de la *Mekhilta de Rabbi Yishmael* nous permettra de définir un ensemble de règles heuristiques que nous combinerons avec d'autres ressources au travers de l'utilisation de la bibliothèque Python *Snorkel* dédiée à la supervision faible. Nous démontrerons ici l'efficacité de cet outil pour résoudre notre problème, ainsi que ses performances dans des cas mêlant des sources plus ou moins fiables. Cette étude prépare le terrain en vue de l'élaboration d'un modèle qui généralisera la détection de citations bibliques à l'ensemble de la littérature rabbinique.

Mots clés : littérature rabbinique, hébreu, supervision faible, qualité de données, annotation automatique, Snorkel, text reuse, citations, manuscrits, midrash

Introduction

Présentation générale

L'augmentation des performances matérielles ainsi que l'apparition de nouvelles technologies facilitant l'implémentation de systèmes d'apprentissage automatique semblent avoir rendu plus accessible la création de modèles permettant, par exemple, la classification d'images ou de documents. Pour celui qui connaît le langage de programmation Python, des bibliothèques telles que Scikit-Learn ou Keras permettent d'implémenter et d'entraîner un modèle en quelques lignes de codes. Cette facilité masque pourtant une difficulté qui est à la base de tous ces systèmes : la constitution de corpus annotés de qualité.

La disponibilité de données suffisamment bien annotées est essentielle pour l'entraînement de ce type de modèles. En effet, dans le cadre de l'apprentissage supervisé, c'est la qualité des annotations qui va déterminer la fiabilité et la robustesse du modèle. Des annotations incomplètes ou de mauvaise qualité engendreront des erreurs dans les prédictions du modèle. La qualité suffisante des annotations est également nécessaire pour l'évaluation des performances des modèles. Des données de test bruitées vont fausser l'analyse des performances des modèles générés. La phase d'annotation des données constitue donc une étape essentielle du processus de construction de modèles en apprentissage automatique.

Néanmoins, si dans certains domaines de grandes quantités de données sont aujourd'hui accessibles, via internet notamment, le volume de données sera beaucoup plus limité lorsqu'il s'agit de langues peu dotées, et plus encore si l'on s'intéresse à des périodes pour lesquelles il ne reste que peu de témoignages écrits disponibles. C'est le cas pour l'étude de textes écrits dans des langues anciennes. Ce problème se retrouve aussi lorsqu'il s'agit d'étudier des manuscrits médiévaux de littérature rabbinique. Dans ce domaine, il n'y a pas de corpus annoté disponible et directement utilisables pour entraîner un modèle d'apprentissage automatique à détecter les citations bibliques. Le cœur de ce mémoire va donc être d'évaluer différentes méthodes en vue d'automatiser la création d'un tel corpus.

Cette étude constitue une étape préparatoire à l'entraînement d'un modèle plus perfectionné d'apprentissage automatique. Nous étudierons une des techniques permettant d'annoter des données sans avoir à faire appel à des annotateurs humains : la *weak supervision*, ou supervision faible. Pour cela, nous allons tester *Snorkel* une bibliothèque Python conçue pour automatiser la création de données d'entraînement. L'ambition d'un tel outil est de faciliter la création de jeux de données fiables à partir de sources bruitées. Ce point de départ peut paraître surprenant puisque, comme

nous l'avons vu, la qualité des modèles dépend souvent de la qualité des données d'entraînement. Plusieurs questions émergent alors : dans quelle mesure un tel outil peut s'avérer efficace lorsqu'il est confronté à des données très bruitées ? Est-il nécessaire d'établir un équilibre entre signaux bruités et non-bruités pour obtenir un modèle performant ? Plus largement, la question centrale est de savoir si un outil comme *Snorkel* apporte réellement un avantage significatif par rapport aux approches heuristiques sans supervision faible.

L'étude d'un *midrash* daté des environs des IIe-IIIe siècles de notre ère, la *Mekhilta de Rabbi Yishmael* va nous permettre d'apporter une réponse à cette interrogation. La *Mekhilta de Rabbi Yishmael* est un commentaire rabbinique du livre de *L'Exode*. À ce titre, il contient un nombre important de citations bibliques, plus ou moins difficiles à repérer. Des méthodes heuristiques pour identifier les citations bibliques existent. Néanmoins, ces méthodes peinent à détecter certaines citations, notamment les plus courtes. Lorsqu'elles y parviennent, cela entraîne une importante diminution de la précision, autrement dit une hausse des faux positifs qui s'accompagne d'un bruit plus important dans les résultats.

Il s'agira donc ici de voir si la supervision faible, en combinant la faible précision des annotations produites par ces méthodes avec d'autres annotations plus précises obtenues avec des règles, permettra d'améliorer l'équilibre entre la précision et le taux de détection des citations courtes.

Notre étude va démontrer l'utilité de *Snorkel* lorsqu'il s'agit d'assembler des sources bruitées (règles imprécises, annotations incomplètes, etc...), mais aussi dans la combinaison avec des sources plus fiables. Pour cela, nous testerons différentes configurations qui vont démontrer l'efficacité de *Snorkel*, en particulier dans l'association de signaux de différents niveaux de qualités. Les expériences réalisées vont démontrer que l'association dans *Snorkel* de sources fiables avec des sources bruitées permet d'améliorer la précision sans faire baisser le rappel.

Nous espérons que ce travail servira aux ingénieurs et chercheurs en Traitement Automatique des Langues (TAL) confrontés eux-aussi au manque de données annotées pour l'apprentissage automatique.

Plan de lecture

Ce travail s'articule en deux grandes parties. Dans l'*état de l'art*, nous commencerons par préciser le concept de réutilisation de texte (*text reuse*) afin de mieux comprendre les difficultés spécifiques à la détection des citations bibliques dans la littérature rabbinique. Cela nous amènera à analyser le fonctionnement de l'outil actuellement le plus performant dans la détection de citations bibliques en hébreu : le *Citation finder* développé par l'équipe de DICTA (*Israel Center for Texts Analysis*). Dans un second temps, nous explorerons les différentes techniques disponibles pour automatiser l'annotation lorsqu'on a pas ou peu de données d'entraînement et montreront pourquoi nous avons choisi la supervision faible.

Dans la partie *Expériences*, nous commencerons par présenter notre corpus, la *Mekhilta de Rabbi Yishmael*, qui est composée de plusieurs versions différentes, ma-

nuscrites et éditées. Ensuite, nous exposerons les résultats de l'exploration de ce corpus, préliminaires au développement des méthodes heuristiques qui nous serviront à alimenter *Snorkel*. Enfin nous détaillerons les expériences réalisées avec *Snorkel* et montrerons en quoi ces expériences nous ont permis d'améliorer la précision et le rappel dans notre tâche de détection des citations dans la *Mekhilta de Rabbi Yishmael*.

Le projet MIDRASH

Ce travail est réalisé dans le cadre du projet ERC SYNERGIE MIDRASH *Migrations of Textual and Scribal Traditions via Large-Scale Computational Analysis of Medieval Manuscripts in Hebrew Script* porté par Daniel Stökl Ben Ezra (Directeur d'études à l'EPHE – PSL) et Judith Olszowy-Schlanger (Directrice d'études à l'EPHE – PSL / Université d'Oxford), en collaboration avec Nachum Dershowitz (Tel Aviv University) et Avi Schmidman (Bar-Ilan University), ainsi que la Bibliothèque nationale d'Israël et l'Université de Haïfa.

Le projet MIDRASH est un projet européen d'une envergure inédite, qui entend s'intéresser à un fond de manuscrits et de documents anciens en écriture hébraïque fort de millions de pages. En associant des spécialistes en Humanité et des informaticiens spécialistes en vision par ordinateur et Traitement Automatique des Langues (TAL), ce projet a notamment pour ambition de développer des technologies informatiques innovantes facilitant l'analyse linguistique, paléographique, philologique ou historique massive de ces données.

À l'origine de ce projet se trouve le projet *Escriptorium*, infrastructure numérique open source du programme *Scripta PSL*. Cette infrastructure a pour but de permettre la transcription automatique de documents manuscrits ou imprimés. Elle est à la base du projet MIDRASH puisque c'est cet outil qui va permettre de numériser les manuscrits et documents anciens analysés dans le projet.

Première partie

État de l’art

Chapitre 1

Text Reuse et Manuscrits rabbiniques

1.1 *Text Reuse*

1.1.1 Définition générale

Le concept de *Text Reuse*, réutilisation de texte en français, regroupe des pratiques très variées. Dans [Moritz et al., 2016] la réutilisation de textes est définie comme : *citing, copying or alluding text excerpts from a text resource to a new context.* Sous une autre forme [Moritz and Büchler, 2017] proposent : *Text reuse is a common way to transfer historical texts. It refers to the repetition of text in a new context and ranges from near verbatim (literal) and para-phrasal reuse to completely non-literal reuse (e.g., allusions or translations).* On trouve une définition similaire chez d'autres auteurs [Lee, 2007] : *Textreuse is the transformation of a source text into a target text in order to serve a different purpose.*.

On peut extraire de ces définitions quelques traits communs aux réutilisations de textes :

— Répétition de tout ou partie d'un texte d'un contexte dans un autre
— Nécessité d'un texte source et d'un texte cible
— Variétés des types de *Text Reuse* du plus littéral au moins littéral

On comprend que les deux traits saillants de la réutilisation de textes sont la relation qui est établie entre deux textes (1) et la plus ou moins grande similitude entre ces deux textes (2). En ce sens la détection du *Text Reuse* pourrait se traduire en termes informatiques comme un problème de distance entre un texte source et un texte cible. Nous verrons plus loin que ce n'est pas aussi simple.

Pour conclure cette approche très général, on peut ajouter avec [MacLaughlin et al., 2021] qu'il convient également de différencier les réutilisations de textes locales, c'est à dire lorsque le texte source et le texte cible contiennent tous deux des parties de textes similaires, mais entourées par du texte qui n'a rien à voir, des réutilisation globales, qui ne sont pas analysées dans l'article, mais correspondent a priori à la réutilisation de l'ensemble du texte source pour construire

le texte cible. Les exemples de ce type de réutilisation de texte global pourraient correspondre à la traduction complète d'une oeuvre dans une nouvelle langue ou à l'établissement de différentes versions d'un même texte.

Cette première étape nous donne une vision très générale de ce qu'est la réutilisation de texte. Mais, comment comprendre la variété des types de réutilisation ?

1.1.2 Typologie

[MacLaughlin et al., 2021] tire de l'état de l'art quelques exemples liés à différents contextes de production :

— Dans le cadre juridique : la réutilisation par un auteur de ses propres textes pour établir une nouvelle version [Wilkerson et al., 2015]
— Dans le cadre journalistique : la reprise d'éléments d'interviews ou de dépêches via la citation ou la paraphrase [Niculae et al., 2015]
— Dans le cadre académique : le résumé dans un article scientifique de la thèse d'un autre auteur [Qazvinian and Radev, 2010]

[Franzini, 2016] propose une typologie de ces différents types de réutilisations en se basant sur la différenciation entre réutilisations d'ordre sémantique (allusion, paraphrase, résumé, etc..) et les réutilisations d'ordre syntaxique, telles que les citations. Cette typologie est résumée en 1.1.

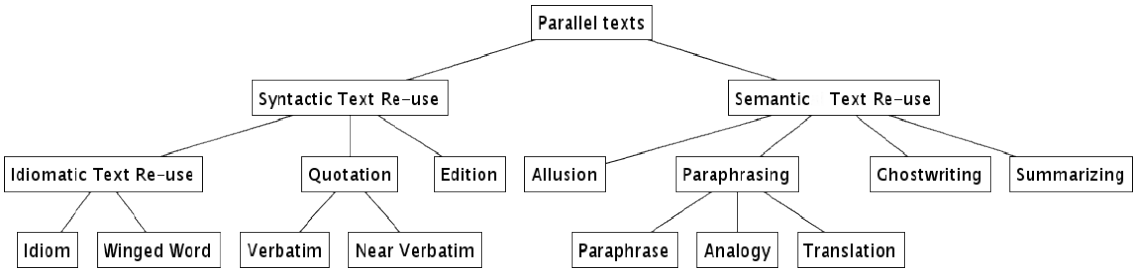

FIGURE 1.1 – Typologie des réutilisations de textes selon Büchler

Maintenant que nous voyons à quoi correspond la réutilisation de textes de manière générale, nous allons nous intéresser à la réutilisation de textes dans le cadre plus spécifiques des humanités numériques, particulièrement lorsqu'il s'agit de textes anciens

1.1.3 Le *Text Reuse* dans les humanités numériques

[Franzini, 2016] prolonge l'analyse en ajoutant que la réutilisation de textes, lorsqu'il s'agit de textes anciens, est souvent plus complexe. Pour expliquer ce phénomène, il apporte plusieurs explications :

— Les citations n'étaient pas indiquées explicitement avec des guillemets ou autres indications typographiques.
— Les langues anciennes étaient moins standardisées que les langues modernes
— Les variations d'une copie à l'autre d'un manuscrit peuvent aussi être dû à l'intervention, volontaire ou accidentelle, des scribes.
— Parfois le texte source n'existe plus, ce qui rend plus difficile l'identification des parties réutilisées.

L'automatisation du processus d'identification des réutilisations de textes va être rendue plus complexe du fait de ces contraintes. L'identification des réutilisations va en effet nécessiter des connaissances pointues concernant le contexte historique, les sources ainsi que les langues dans lesquelles elles sont écrites. L'expert sollicité va donc devoir être tout à la fois spécialiste d'histoire des textes, linguiste et codicologue.

[Franzini, 2016] donne de nombreux exemples de réutilisation de textes anciens. Prenons celui-ci : deux textes en latin du IXe et du XIIe siècle.

Voici le premier (Wolfhard de Herrieden au IXe siècle) :

*Quo **audito**, vir illustris a loco quo **sederat** illico **prosilivit**, et animo consternatus, quare ad ostium **nobilis** persona et Omnipotentis ancilla constiterit, requisivit. Illa vero non sine causa se venisse asseruit. Tum vir memoratus cum omni veneratione **eam in domo** excepit, **et omnem** famulatum debitae servitutis **exhibuit**.* . (Ex Wolfhardi Haserensis miraculis s. Waldburgis Monheimensibus, éd. Oswald Holder-Egger (MGH SS 15,1, Hanovre 1887) vers 3 p. 540)

Voici le deuxième (Adelbert de Heidenheimin au XIIe siècle) :

*Tunc paterfamilias **audito** nomine Walpurgae de **sede** sua **prosiliuit** et miratus tam **nobilis** et tam sanctae virginis adventum, studiossime totam domum emundavit et omnibus quae ocolus sanctitatis eius offendere possunt, detersis benignissime **eam in domum** sua, recepit **et omne** humanitatis et devotionis obsequium ei **exhibuit**.* . (Adelbert de Heidenheim, Relatio, éd. Jakob Gretser (Ingolstadt 1617) p. 330-331. Nouvelle édition de Christian Schwaderer

Dans l'exemple ci-dessus, les mots en gras permettent de repérer les similitudes entre les deux textes. Toutefois, on voit que la fréquence de mots communs, même en prenant en compte les transformations morphologiques, n'est pas très élevée. Pourtant, le deuxième texte a bien été confirmé comme étant une réutilisation du premier par un spécialiste du domaine. Comme le note [Franzini, 2016], un néophyte pourra éventuellement repérer quelques similitudes liées à l'utilisation d'un vocabulaire commun dans les deux textes. Néanmoins, seul un expert pourra déterminer qu'il s'agit réellement d'une réutilisation ainsi que la nature de cette réutilisation.

Maintenant que nous comprenons quels sont les enjeux liés à l'identification de réutilisations de textes dans des textes anciens, regardons quelles solutions techniques ont été élaborées en vue d'automatiser l'identification des réutilisations de

textes. Étant donné la grande diversité des types de *Text Reuse*, nous n'aborderons pas l'ensemble des techniques développées pour la détection de citations, mais plutôt celles développées spécifiquement pour l'hébreu rabbinique.

1.2 Les spécificités des manuscrits anciens en hébreu Rabbinique

Les manuscrits en hébreu rabbinique partagent les spécificités des autres manuscrits telles que nous les avons décrites plus haut : pas d'indications typographiques pour les citations, variabilité de la langue et modifications dues au travail des scribes. Néanmoins, l'hébreu rabbinique possède également des spécificités qui lui sont propres.

Tout d'abord, on trouve des spécificités liées aux caractéristiques précédentes. Par exemple, l'hébreu rabbinique possède une variabilité orthographique plutôt importante, comme d'autres langues anciennes, mais cette variabilité s'explique par ses propres caractéristiques.

Ainsi, en hébreu plusieurs systèmes orthographiques cohabitent. Cela est principalement dû à la place particulière des voyelles dans l'alphabet hébraïque. Traditionnellement, l'alphabet hébraïque est considéré comme un abjad, c'est à dire un alphabet n'utilisant que des consonnes. Les voyelles n'y sont donc pas originellement symbolisées. Néanmoins, certaines lettres vont progressivement être utilisées, du fait de leurs caractéristiques hybrides, pour symboliser les voyelles. C'est le cas du yod (י), du wav (ו), du aleph (א) et du hé (ה), que l'on considère parfois comme des semi-voyelles. En effet, le י, peut à la fois être utilisé comme consonne [j] et comme voyelle [i]. Le ו peut être utilisé comme consonne [w], ou comme voyelle [o] ou [u]. Les utilisations des lettres א et ה comme voyelles est moins évidente. Le א représente à l'origine un coup de glotte, ou consonne occlusive glottale [ʔ]. Il va donc pouvoir servir de support à tout type de vocalisation. Il va néanmoins être plutôt utilisé pour symboliser la voyelle [a]. De même, le ה symbolise la consonne fricative glottale sourde [h] et peut donc porter tout type de voyelle. En tant que voyelle, il va essentiellement être utilisé en fin de mot pour symboliser les sons [a], [e] ou [o].

Lorsqu'elles sont utilisées comme voyelles, ces lettres sont appelées des *Matres Lectionis*, ou mère de lecture en français. De leur usage découle deux systèmes d'écriture de l'hébreu : l'écriture pleine utilisant les *Matres Lectionis*, כתיב מלא (*Ktiv malé*) en hébreu et l'écriture défective, ou כתיב חסר (*ktiv hasar*), ne les utilisant pas. Dans un même texte ou ensemble de textes, un même mot pourra apparaître sous ses formes pleines et défectives.

Un autre phénomène spécifique à l'hébreu rabbinique est l'utilisation plus ou moins intensive d'abréviations. C'est à dire l'utilisation d'une forme tronquée d'un mot ou d'une expression. C'est le cas par exemple de l'expression תלמוד לומר (*Talmud Lomar*), très fréquente dans la littérature rabbinique, qui possède même plusieurs types d'abréviations, du plus réduit au moins réduit, comme on peut le voir sur la figure 1.2.

L'abréviation pourra être utilisée dès la première occurrence du terme, ce qui

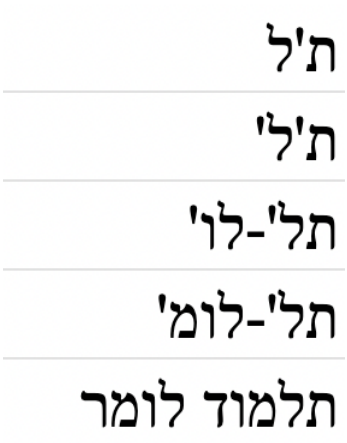

FIGURE 1.2 – l'expression Talmud Lomar : formes pleines et abrégées

n'empêche pas que le mot pourra aussi apparaître sous sa forme non abrégée, voir même avec des variations orthographiques liées à l'utilisation des *Matres Lectionis*. Les première lignes tirées de l'un des manuscrits de la *Mekhilta* visibles sur la figure 1.3 permet de se rendre compte de l'importance du phénomène. les apostrophes entourées en rouge y symbolisent les abréviations :

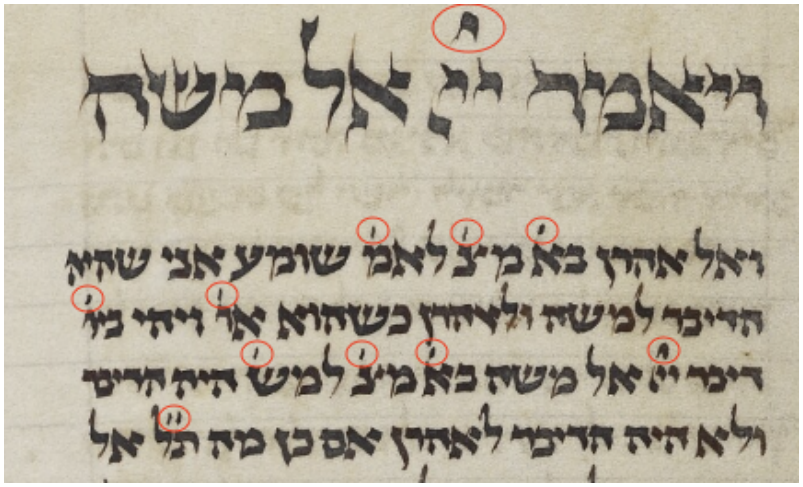

FIGURE 1.3 – Première page de la Mekhilta d'Oxford

1.3 Le *Citation Finder* du projet DICTA

L'équipe du *Israel Center for Texts Analysis* (DICTA) a développé plusieurs outils d'analyse de textes en écriture hébraïque. Parmi ceux-ci, on trouve des outils permettant de vocaliser les textes et d'en faire l'analyse morphologique, de faire des recherches dans la Bible et dans la littérature rabbinique ou de résoudre les problèmes liés aux abréviations et aux variations orthographiques.

Parmi ces outils, le *Citation Finder* [Shmidman, 2022] nous intéresse plus spécifiquement puisqu'il permet de retrouver des citations bibliques dans tous types de textes en hébreu, que ce soit des citations exactes ou des reformulations. Pour

cela, l'algorithme va commencer par décomposer le texte source et le texte cible en unités minimales (*Hashes*). Ces *hashes* représentent les mots en ne conservant que les deux lettres les moins fréquentes. Ce processus de *hashing* va éliminer la majorité des *matres lectionis* et des préfixes, facilitant ainsi la prise en compte des variations orthographiques et morphologiques.

Ensuite, l'algorithme va effectuer des paires à partir des trigrams. Par exemple, pour un trigram donné ABC, l'algorithme va considérer les paires AB, BC et AC. Cela va permettre de détecter les citations même lorsqu'il y a des inversions ou des insertions. Il est important ici de noter que les citations d'un seul mot ne sont pas prises en compte. En effet, en utilisant un algorithme basé sur la similarité, tel que celui de Dicta, de nombreux mots seraient détectés comme citation. De plus, comme l'explique Shmidman à partir de l'analyse de [Einat-Nov, 2013], si l'on considère qu'un seul mot peut constituer une citation ou une allusion biblique, les auteurs ne feraient jamais usage de ce mot sans en invoquer la référence biblique, ce qui en rendrait au final l'utilisation impossible. Nous reviendrons sur ce problème au chapitre 3

Le processus de *hashing* permet de détecter un grand nombre de parallèles. Néanmoins, en comparant nos textes sources et cibles en se basant uniquement sur cette représentation, l'algorithme va produire de nombreux faux positifs. Par exemple, certains bigrams ont une fréquence très élevée dans la Bible, voire même dans la langue hébraïque elle-même, ce qui rend l'identification plus compliquée. Il faut donc que l'extrait soit suffisamment singulier pour pouvoir être distingué.

Le critère de rareté va alors servir à nettoyer les faux-positifs de l'ensemble des candidats-citations produits par le système. Pour qu'une combinaison de *hashes* présente à la fois dans les deux textes soit considérée comme une citation, il faut que cette combinaison soit suffisamment rare. L'interface en ligne offre d'ailleurs la possibilité de faire varier le seuil (*threshold*) et ainsi faire un choix entre précision et rappel. L'utilisateur peut choisir de rechercher uniquement des combinaisons très rares, ce qui va augmenter la précision du système en réduisant le nombre de faux positifs, tout en augmentant le nombre de faux négatifs, c'est à dire des citations réelles non détectées. Il peut aussi choisir de réduire cette sensibilité à la rareté et ainsi augmenter la quantité de citations détectées (vrais positifs). Néanmoins le résultat obtenu contiendra également un nombre plus important de faux positifs.

L'extrait présenté en 1.4 illustre les différents types de réutilisations prises en compte par le *Citation Finder* de Dicta (interface web). La table 1.1 explicite les transformations rencontrées.

1.4 Conclusion

Les paragraphes précédents montrent toute la difficulté que représente la détection de réutilisations de textes lorsqu'il s'agit de manuscrits, et plus particulièrement de manuscrits rabbiniques. Une autre approche serait d'utiliser des méthodes basées sur l'apprentissage automatique. Seulement, l'absence de corpus annoté disponible nous empêche pour l'instant de faire appel à ces techniques.

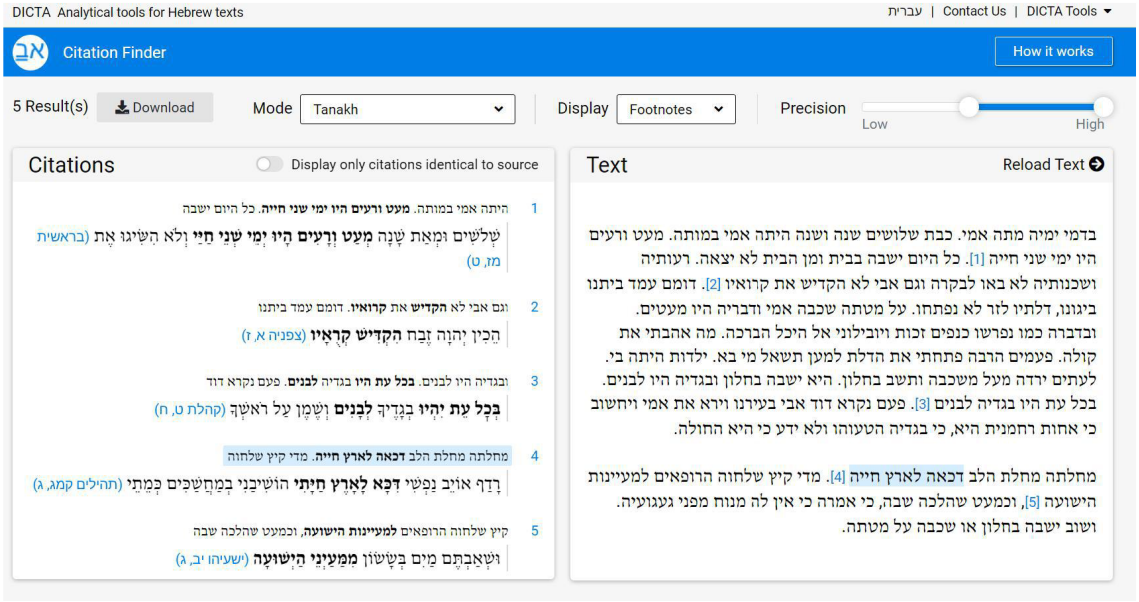

FIGURE 1.4 – Interface de Dicta online avec un extrait de l'auteur israélien Shaï Agnon en exemple

Modification	Agnon	Citation biblique
Flexion	מעט ורעים היו ימי שני חיי	מעט ורעים היו ימי שני חייה
Insertion	הקדיש את קראין	הקדיש קראין
Flexion	בכל עת היו בגדיה לבנים	בכל עת יהיו בגדיך לבנים
Flexion	דכאה לארץ חייה	דכא לארץ חיתי
Prep. + flexion	למעיינות הישועה	ממעיני הישועה

TABLE 1.1 – types de réutilisations détectées par Dicta

Comme nous l'avons vu précédemment, le repérage de réutilisations de textes nécessite des connaissances expertes dans des domaines aussi variés que l'histoire, la linguistique ou la codicologie. Concernant notre problème, l'absence de corpus annoté, la taille importante du corpus concerné et la nécessité de devoir passer par des experts pour une éventuelle annotation manuelle nous invite à explorer d'autres méthodes.

Nous allons donc maintenant explorer les différentes techniques permettant d'automatiser le processus d'annotation, en limitant le plus possible le recours à des annotateurs humains.

Chapitre 2

Techniques d'annotation automatisées

2.1 Introduction

L'annotation, ou labellisation, de données brutes est l'objectif que se fixent toutes les méthodes utilisées en TAL. Que l'on parte de données annotées ou non, l'objectif va toujours être de mettre en place des modèles capables de généraliser les informations contenues dans les données d'entraînement. Ce qui va changer d'un projet à l'autre et déterminer le type de technique à utiliser va essentiellement être la quantité de données annotées disponibles au départ du projet, ainsi que les ressources allouées. Le schéma en 2.1 montre différentes techniques d'annotation classées en fonction de la quantité de données annotées disponibles et du mode d'obtention des annotations.

Dans ce chapitre, l'idée va être dans un premier temps de présenter ces différentes méthodes, pour ensuite voir en quoi elles pourraient être adaptées ou non à notre problème de détection des citations dans un corpus de littérature rabbinique. Nous étudierons dans un premier temps les techniques traditionnelles d'apprentissage automatique supervisé et non-supervisé. Ensuite nous nous intéresserons à la nébuleuse des techniques d'apprentissage semi-supervisées. Puis nous présenterons les techniques d'apprentissage actif (*activ learning*) et d'apprentissage par transfert (*transfer learning*). La supervision faible sera exposée en fin de présentation car, comme nous le verrons, celle-ci permet de combiner les différentes méthodes citées précédemment. Nous verrons alors pourquoi cette méthode s'adapte particulièrement bien à notre projet

2.2 Apprentissage automatique supervisé et non-supervisé

L'apprentissage supervisé regroupe toutes les méthodes qui vont, à partir de données entièrement annotées, chercher à construire des modèles capables de généraliser pour annoter des ensembles de données plus grands. Il regroupe des méthodes très diverses, allant de la régression logistique aux réseaux de neurones, toutes ces méthodes ayant en commun d'utiliser des données annotées pour l'en-

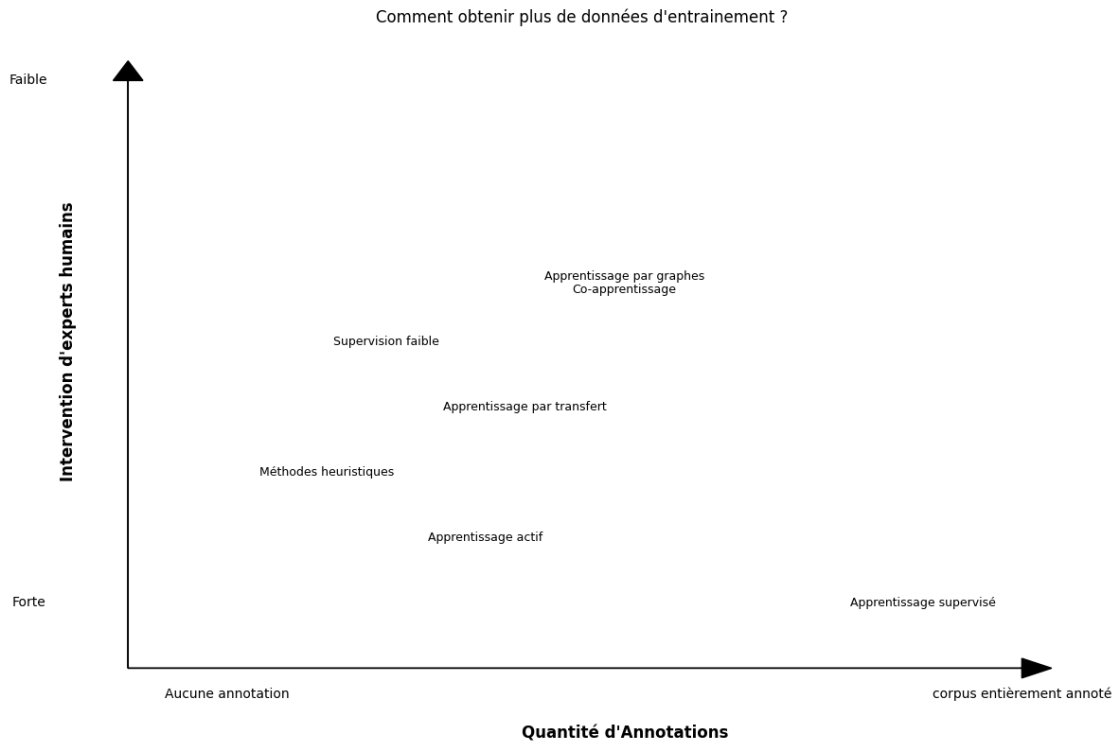

FIGURE 2.1 – Les différentes techniques d'annotation en fonction de la quantité de données annotées disponibles et du mode d'obtention des annotations

traînement de leur modèle. Les techniques d'apprentissage non-supervisé vont faire la même chose, mais à partir de données non-annotées. C'est à dire que les méthodes non-supervisées vont chercher des patterns, des récurrences statistiques, dans des données non-annotées afin de pouvoir catégoriser de nouvelles données.

Dans le cadre de notre projet, l'apprentissage supervisé est inenvisageable à ce stade, étant donné que nous n'avons pas du tout de données annotées fiables. De plus, en absence de données annotées, il serait aussi très difficile de pouvoir évaluer la qualité des prédictions.

L'apprentissage non-supervisé pourrait, au premier abord, sembler être une bonne idée. Il est néanmoins à exclure car souvent basé sur le regroupement (*clustering*) de données se basant sur des caractéristiques communes. Le modèle généré risque donc de détecter de nombreux phénomènes dans les données sans que l'on puisse savoir ce qui correspond ou non à des citations. En général, les méthodes d'apprentissage non-supervisées sont plutôt utilisées pour faire émerger des patterns dans les données, plutôt que pour détecter un phénomène particulier. Dans son champ d'application on peut citer le *topic modelling*, qui vise à *diviser un document en segments thématiquement cohérents* [Benamar, 2020], ou les plongements lexicaux (*embeddings*) tels que *word2vec* ou *glove*.

2.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé regroupe toutes les techniques qui vont se servir de données étiquetées et non-étiquetées. Généralement le jeu de données proposé

contient peu de données étiquetées et beaucoup de données non étiquetées. Étant donné le coup élevé de l'annotation par des humains, ce type de situation est relativement répandu. La question est alors : comment le modèle va t'il apprendre à partir des données non-annotées ?

L'apprentissage semi-supervisé regroupe de nombreuses méthodes, nous en détaillerons ici deux parmi les plus connues : l'auto-apprentissage, le co-apprentissage. La plupart des explications sont tirées de et de [Maamatou, 2017]

2.3.1 L'auto-apprentissage

Le fonctionnement de l'auto-apprentissage est plutôt simple. Il va s'agir d'entraîner un premier modèle en utilisant uniquement les données étiquetées. Puis ce modèle va être utilisé pour créer de nouvelles étiquettes en effectuant des prédictions sur les données non-étiquetées. Ce modèle va alors utiliser sur sa propre fonction de décision pour mettre à jour l'ensemble d'apprentissage en déterminant quelles prédictions sont fiables. Les données nouvellement incorporées dans l'ensemble d'entraînement vont alors servir pour reformer un nouveau modèle avec plus de données. Le processus va être répété de manière itérative jusqu'à obtenir les performances souhaitées.

Le risque le plus probable avec ce genre de méthode, est que le modèle intègre des prédictions fausses qui vont se renforcer à chaque nouvelle itération du processus. Dans ce cadre, le paramétrage de la fonction de décision, la définition du bon seuil de confiance et l'ajustement des critères d'arrêt des itérations vont être des éléments essentiels à la réussite d'une stratégie basée sur l'auto-apprentissage.

L'auto-apprentissage est une technique qui peut être utilisée seule ou en combinaisons avec d'autres techniques. Par exemple, [Caragea et al., 2015] ont démontré que l'auto-apprentissage permettait d'obtenir des gains de performances significatifs en combinaison avec RoBERTa et des techniques d'augmentation de données en vue de constituer des datasets labellisés à partir de données issues du web

2.3.2 Le co-apprentissage

L'algorithme a été défini pour la première fois par [Blum and Mitchell, 1998]. Comme avec l'auto-apprentissage, le co-apprentissage, ou *co-training*, est basé sur un processus itératif. Seulement cette fois-ci ce seront non pas un mais deux modèles qui vont être entraînés sur les données étiquetées du dataset. Pour cela deux "vues" différentes des données étiquetées vont être utilisées pour chaque modèle, c'est à dire que les mêmes données vont être utilisées mais avec des caractéristiques (*features*) différentes. Ensuite, les deux modèles vont être utilisés pour effectuer des prédictions sur les données non étiquetées. Les prédictions considérées comme fiables par l'un des deux modèles vont être ajoutées à l'ensemble d'entraînement de l'autre modèle. Comme avec l'auto-apprentissage, le processus va être répété plusieurs fois jusqu'à obtenir des performances optimales, c'est à dire à partir du moment où les modèles ne seront plus en mesure de fournir des prédictions fiables.

L'idée est qu'en utilisant des caractéristiques différentes des données pour entraîner les deux classifieurs, on espère compenser les faiblesses du premier

classifieur avec le deuxième. Le risque de propager des fausses prédictions est donc mieux contrôlé par rapport à l'auto-apprentissage. Néanmoins, ce système nécessite d'avoir pour chacun des classifieurs des vues sur les données qui soient suffisamment différenciées, mais tout en restant complémentaires.

Le co-apprentissage est une technique couramment utilisée en TAL, que ce soit pour l'analyse de sentiments [Wan, 2009], pour la classification de *commits* [Lee and Chieu, 2021] ou pour le *topic modeling* [Caragea et al., 2015]

2.3.3 Conclusion sur l'apprentissage semi-supervisé

En comparant les différentes techniques d'apprentissage semi supervisées, on remarque qu'elles ont toutes en commun d'exploiter un volume restreint de données annotées pour effectuer des prédictions sur des données non-annotées au travers d'un processus itératif. Ce processus reste cependant hasardeux, dépendant grandement de la qualité de l'annotation initiale et de la représentation des données.

Dans notre cas, l'utilisation de ce type de méthode ne semble pas adapté car nous n'avions pas en début de projet de données étiquetées avec suffisamment de fiabilité. Le risque serait alors de propager des erreurs de prédiction, augmentant à chaque itération le poids de ces prédictions fausses dans le modèle.

2.4 Apprentissage actif

L'apprentissage actif est une technique d'annotation des données qui va consister à minimiser le plus possible la quantité de données à annoter en ne s'intéressant qu'aux données les plus pertinentes, c'est à dire celles qui vont le plus améliorer le modèle. Il s'agit donc de minimiser le coût de l'annotation tout en maximisant son efficacité sur les performances du modèle. Pour cela, l'apprentissage actif préconise un processus itératif au cours duquel on va commencer par générer un modèle de base à partir d'un ensemble réduit de données étiquetées. Le système va alors déterminer quelles sont les données les plus pertinentes à annoter parmi les données non-étiquetées puis faire appel à un "Oracle" pour étiqueter ces données. En général l'oracle est un humain, mais il peut aussi être une machine.

L'apprentissage actif peut être considéré comme un type d'apprentissage semi-supervisé. Nous avons néanmoins choisis de le présenter à part car, comme on peut le voir sur le schéma 2.1, celui-ci se démarque des autres méthodes d'apprentissage semi-supervisé du fait du recours à un oracle pour annoter les données.

Comme d'autres méthodes, l'apprentissage actif peut-être utilisé en combinaison avec d'autres techniques, telles que des modèles de langue de type BERT. [Yuan et al., 2020] se servent de la fonction de perte de BERT pour déterminer les exemples qui vont être proposés à l'oracle. Pour rentrer plus en détail dans l'état de l'art de l'apprentissage actif, [Zhang et al., 2022] passent en revue les publications récentes en rapport avec l'apprentissage actif et ses différentes applications en TAL.

L'apprentissage actif pourrait être une piste intéressante pour la résolution de notre problème de détection de citations. Nous ne l'avons pas explorée dans le cadre

de ce mémoire du fait de l'absence de données annotées au départ de notre projet. Il pourrait néanmoins être une piste à explorer en combinaison avec le tout nouveau modèle de langue BEREL (BERT Embeddings for Rabbinic-Encoded Languages) développé par DICTA [Shmidman et al., 2022].

2.5 Apprentissage par transfert

L'apprentissage par transfert (*Transfer learning*) consiste à utiliser sur de nouvelles tâches des modèles pré-entraînés sur d'autres tâches. Cette technique fait bien évidemment penser aux modèles de langues pré-entraînés du type BERT ou GPT. L'utilisation de l'apprentissage par transfert reste néanmoins antérieure à l'apparition des *Large Languages Models* (LLM). On la trouve exposée chez [Caruana, 1993]. Plus récemment, [Augenstein et al., 2015] ont proposé une technique d'apprentissage par transfert où l'utilisation d'un classifieur entraîné sur la reconnaissance d'entité nommées permet de reconnaître d'autres types d'entités.

Le modèle de DICTA BEREL [Shmidman et al., 2022] serait parfait pour résoudre notre problème. Seulement, nous aurons besoin de données annotées pour pouvoir entraîner un classifieur à reconnaître les citations bibliques en utilisant BEREL. Cette expérience constituera donc la partie suivante de notre projet, une fois que nous aurons trouvé un moyen efficace de constituer un corpus de données annotées.

2.6 La supervision faible

La supervision faible, ou *weak supervision* est un paradigme qui repose sur l'idée de construire un corpus annoté à partir d'informations imparfaites (règles heuristiques, modèles d'apprentissage automatique, annotations issues du *crowdfunding*, etc...), pour obtenir un résultat relativement plus fiable. Ensuite, un algorithme va être utilisé pour pondérer automatiquement ces différentes sources d'information afin d'en évaluer la fiabilité. Ce type d'approche est particulièrement adapté aux contextes où l'on ne dispose ni de données annotées ni de moyens pour annoter ces données. La supervision faible se présente comme une alternative pour constituer rapidement et efficacement un corpus d'entraînement dans des conditions contraintes.

2.6.1 L'utilisation de la supervision faible dans les recherches récentes

L'observation des dates de parution des publications scientifiques où apparaît le terme *Weak supervision* semble indiquer un développement récent de cette technique, autour de 2015. Cela est dû à plusieurs facteurs. Tout d'abord, le besoin de quantités importantes de données annotées n'a réellement commencé à se faire sentir qu'à partir du développement de techniques récentes nécessitant une forte puissance de calcul et de gros volumes de données. Les réseaux de neurones et les modèles de langues pré-entraînés ont ce type d'exigences. De plus, l'idée étant de répondre à un besoin d'efficacité, l'apparition d'outils clé en main, comme *Snorkel*

[Ratner et al., 2017] ou *Skweak* [Lison et al., 2021], a permis un essor plus important de cette technique.

Néanmoins, l'analyse des articles récents concernant l'utilisation de la supervision faible fait parfois émerger une acception plus large de ce concept. Dans l'un des articles les plus récents [Heck et al., 2022], les auteurs utilisent l'auto-apprentissage pour former un modèle de suivi des dialogues (DST) à reconnaître les informations importantes d'une conversation sans avoir besoin d'étiquettes précises pour chaque exemple. Le point commun entre la supervision faible telle qu'envisagée par *Snorkel* ou *Skweak* et la supervision faible telle que décrite dans cet article est le manque de données annotées au départ. Cependant, les méthodes divergent au sens où les auteurs ne combinent pas différents signaux, issus par exemple de fonctions de labellisation heuristiques ou de dictionnaires. Dans cet article, les auteurs utilisent indifféremment les concepts de *supervision faible* et *d'auto-apprentissage*. La porosité dans la définition des différentes techniques de *weak supervision*, *auto-apprentissage* ou même *apprentissage semi-supervisé* est liée aux points communs entre ces techniques. Elles se rejoignent toutes dans la recherche de solutions dans un contexte contraint par le manque de données annotées. Elle est aussi dû au fait que, loin de s'exclure, ces différentes techniques peuvent être combinées.

Dans un article de 2021 [Karamanolakis et al., 2021] combinent supervision faible et auto-apprentissage pour réaliser une tâche de classification de textes. Pour cela ils utilisent deux modèles. Le premier, le modèle "étudiant", va être alimenté avec des données non-étiquetées. Au travers d'un processus itératif d'auto-apprentissage, ce modèle va faire des prédictions qui vont être évaluée par le second modèle, dit "enseignant". Le modèle "enseignant" va aider l'étudiant à ajuster ses prédictions au moyen de règles heuristiques.

2.6.2 Deux outils clé en main : *Snorkel* et *Skweak*

Snorkel et *Skweak* proposent une approche similaire. Il s'agit de permettre à l'utilisateur de combiner différentes sources (méthodes heuristiques, étiquettes bruitées issues d'annotateurs humains ou non, etc...) qui vont ensuite être débruitées par un algorithme probabiliste.

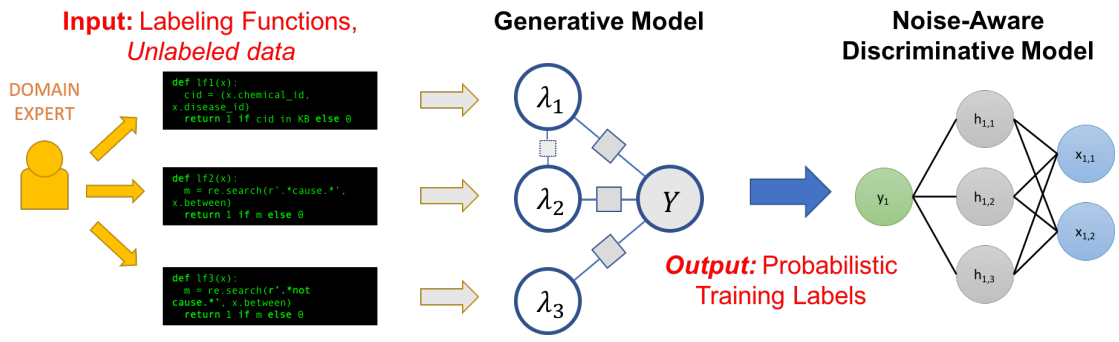

FIGURE 2.2 – Chaine de traitement dans Snorkel

La figure 2.2 montre la chaîne de traitement dans *Snorkel*. Un expert du domaine va créer des fonctions de labellisation (LFs) qui vont être appliquées aux données.

Ensuite, le modèle va être utilisé pour évaluer la fiabilité des LFs et pondérer leurs sorties en cas de conflit. Ce modèle va alors produire un ensemble d'étiquettes probabilistes. Nous verrons que *Snorkel* propose cette approche probabiliste, mais que, en pratique, il est aussi possible d'utiliser un modèle basé sur le vote majoritaire pour déterminer la précision des LFs.

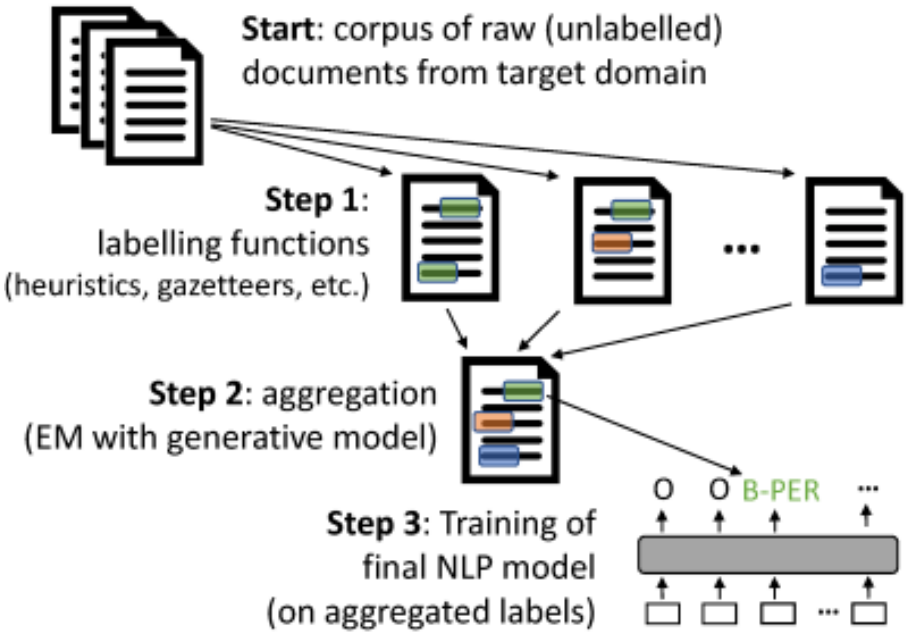

FIGURE 2.3 – Chaine de traitement dans Skweak

La figure 2.3 montre une chaine de traitement similaire avec *Skweak*. Cependant *Skweak* se différencie de *Snorkel* sur quelques points clés qui vont déterminer notre choix. Tout d'abord, *Skweak* utilise un modèle d'agrégation différent, basé sur un algorithme d'expectation-maximisation (EM). Cet algorithme reste un algorithme probabiliste, mais différent de celui proposé par *Snorkel*. Là où *Skweak* se démarque réellement, c'est par sa spécialisation dans le domaine du TAL. En effet, la chaîne de traitement de *Skweak* est adossée à *Spacy*, ce qui permet d'enrichir le traitement du modèle de *Skweak* de toutes les fonctionnalités de *Spacy*. Cette caractéristiques a déterminé notre choix à se porter sur *Snorkel* étant donné que *Spacy* n'est pas nativement adapté au traitement de l'hébreu rabbinique.

2.7 Conclusion

Parmi toutes les techniques exposées dans ce chapitre, la supervision faible nous est apparue comme un bon choix. Démarrer le projet avec un corpus de données non-annotées nous a incité à explorer dans un premier temps des méthodes heuristiques. Plus récemment, l'accès aux annotations générées par le *citation finder* de DICTA nous a conforté dans l'idée d'utiliser un système permettant de combiner nos règles avec les annotations de DICTA. Nous avons ensuite effectué quelques tests avec *Snorkel* et *Skweak*. Le fait que *Skweak* ne puisse se passer de *Spacy*, qui n'a pas de modèle adapté à l'hébreu rabbinique, nous a incité à utiliser *Snorkel*.

Les autres techniques présentées pourrait néanmoins être intéressantes à envisager une fois que nous aurons un peu plus de données annotées, notamment l'utilisation du modèle de langue adapté à l'hébreu rabbinique BEREL.

Deuxième partie

Expériences

Chapitre 3

Le corpus : La Mekhilta de Rabbi Yishmael

3.1 Introduction

Notre corpus est constitué de différentes versions, manuscrites et éditées, de la *Mekhilta de Rabbi Ishmael*, que nous dénommerons simplement *Mekhilta* dans le reste de ce rapport. La *Mekhilta* est un texte initialement rédigé durant la période tannaïtique, qui s'étale de l'an 0 à 200 et correspond à l'ère durant laquelle s'est constituée la littérature ayant servi de base à la compilation de la *Mishna* dans les siècles suivants. L'ensemble de la littérature rabbinique est aussi dénommée Loi orale, par opposition avec la Loi écrite, qui désigne plus spécifiquement le *Tanakh* (*Torah*, *ktouvim*, *nevihim*), aussi dénommé Bible hébraïque. L'ensemble des enseignements englobés dans la loi orale étaient originellement transmis oralement avant leur compilation à l'écrit à partir du IIe siècle, ce qui explique cette dénomination.

La littérature rabbinique de cette époque est principalement constituée de *midrashim*, c'est à dire de commentaires du *Tanakh*. Ces commentaires sont généralement classifiés en deux types, les *midrashim* halakhique et les *midrashim* haggadiques. Les *midrashim* halakhiques sont consacrés à l'approfondissement des aspects du *Tanakh* liés à la loi juive, aussi appelée *halakha* en hébreu. Les *midrashim* haggadique, sont plutôt destinés à approfondir et éclaircir les récits (*haggadot* en hébreu) bibliques en en extrayant les aspects moraux ou philosophiques.

La *Mekhilta* est un *midrash* halakhique consacré au deuxième livre de la Bible hébraïque, שמות (*Chemot* - Les noms), aussi appelé *L'Exode* en français. La Mekhilta couvre 12 des 40 chapitres de *L'Exode* : 12, 1-13, 19 ; 31, 12-17, 35, 1-3. Elle est originellement divisée en 9 traités (*massekhtot*), subdivisés en 82 paragraphes (*parashiot*) :

1. *Pisha* (Ex. 12, 1 s.), analyse la *parasha Bo* et concerne l'offrande pascale et la sortie d'Egypte
2. *Beshallah* (13, 17s.) concerne le passage de la mer des joncs
3. *Shirota* (15, 1 s.) concerne le *Cantique de la mer*
4. *Wa-Yasa'* (15, 22 s.) la suite, jusque la tombée de la manne
5. *Amalek* (17, 8 s.) concerne la guerre contre Amalek et le conseil de Jéthro

6. *Ba-kodesh* (19, 1 s.) concerne le don de la loi au mont Sinaï
7. *Neziqin* (21, 1 s.) concerne les lois des dommages dans la première partie de la *parasha Mishpatim*
8. *Kaspa* (22,24 s.) concerne les autres lois dans ce qui reste de la *parasha Mishpatim*
9. *Shabbeta* (31, 12-17 ; 35, 1-3) analyse les aspects des *parashiot Ki Tisa et Vayakhel* concernant les lois de *Shabbat*

Cette répartition originelle peut varier selon les versions, même si elle reste la plus commune [Stemberger and Bockmuehl, 1996]

Des manuscrits présentant diverses variations sont aujourd'hui à notre disposition, faisant de l'origine et de l'évolution du texte de la *Mekhilta* un enjeu important pour la recherche.

3.2 Versions manuscrites et éditées

Les trois manuscrits encore accessibles [Nelson, 1999], [Stemberger and Bockmuehl, 1996] :

— Le manuscrit d'Oxford (*Bodleian Library MS Marshall Or. 24*) daté de la fin du 13e siècle (1291) - Complet
— Le manuscrit de Munich (*bayerisch staatsbibliothek Cod.Heb.117*), daté du milieu du 15e siècle (1435) - Complet
— Le manuscrit du Vatican (*biblioteca apostolica vaticana Ebr 299.6*)- Incomplet

Suite à l'invention de l'imprimerie, plusieurs éditions de la *Mekhilta* sont apparues :

— L'édition Princeps imprimée en 1515 à Constantinople
— L'édition de Venise, imprimée en 1545
— L'édition de J-H Weiss, Vienne, 1865
— L'édition de M. Friedmann, Vienne, 1870

Les deux versions modernes éditées les plus courantes actuellement sont, d'après [Stemberger and Bockmuehl, 1996] :

— L'édition de H.S. Horovitz et I. A. Rabin, datée de 1931 et principalement basée sur l'édition de Venise (1545)
— L'édition de J. Z. Lauterbach, datée de 1933, basée sur les manuscrits et les éditions précédentes

L'édition de Horovitz et Rabin (HR) présente l'intérêt de référencer une partie des citations parmi les plus longues et les plus évidentes.

Il existe aussi deux version en ligne qui, elles aussi, indiquent les citations mais sans forcément préciser les références :

— L'édition proposée par le site web collaboratif https ://www.sefaria.org/
— L'édition proposé par l'académie nationale de la langue hébraïque sur le site https ://maagarim.hebrew-academy.org.il/

3.3 Données à notre disposition

Dans le cadre de notre projet, nous avons pu récupérer les données issues des manuscrits d'Oxford (Oxf) et de Munich (Mun), ainsi que celles issues de l'édition de Venise (Ven) et de Horovitz et Rabin (HR). Ces ouvrages ont tous été numérisés grâce à l'outil *Escriptorium*, développé par les équipe du projet *Scripta*.

3.3.1 Données issues des manuscrits

Ces données, au format tabulaire et XML-TEI, regroupent des informations sur le texte en lui-même, mais aussi des informations liées à la mise en page (position des mots, taille du texte, etc..). Ces informations sont très importantes pour reconstituer le texte numériquement, car il n'est pas toujours linéaire. Dans le cas des manuscrits, il peut contenir des ajouts ou des suppressions effectuées par différents scribes, comme on peut le voir dans l'extrait de la Mekhilta d'Oxford proposé sur la figure 3.1

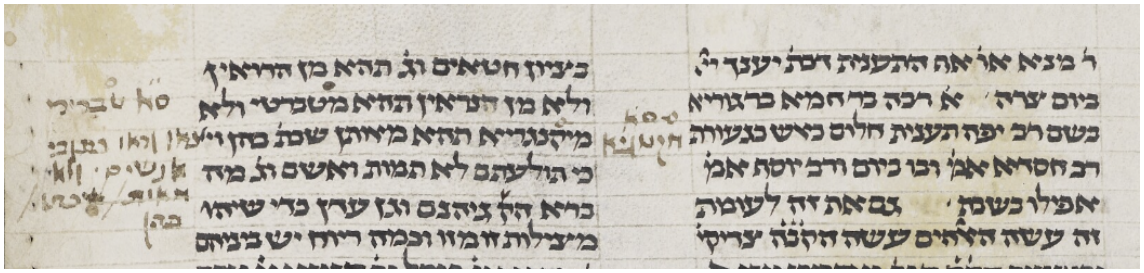

FIGURE 3.1 – Oxford, Bodleian Library MS. Marshall Or. 24 : https ://digital.bodleian.ox.ac.uk/objects/ddff16f5-85a6-45be-abf0-3385c2a2cadc/

La force de l'outil de numérisation *Escriptorium* réside dans sa capacité à segmenter les différentes zones de la page en "régions" et en "segments", ce qui permet d'étiqueter toutes ces zones séparément et ainsi différencier, par exemple, les paragraphes des entêtes et pieds de pages, ou d'images telles que les *line fillers*.

3.3.2 Données issues de l'édition de Horovitz et Rabin

Concernant HR, nous avons à notre disposition une numérisation de l'ouvrage au format *tsv*. Le fichier contient 2 colonnes, une pour chaque token et une avec un identifiant unique. Ultérieurement nous avons aussi ajouté une colonne qui permet de repérer les tokens correspondants à des références bibliques. En effet, l'intérêt principal de cette édition est quelle indique les références bibliques. Dans la première partie de l'ouvrage ces références sont indiquées entre parenthèses à la suite des citations , ce qui permet d'identifier la fin mais pas le début de la citation, comme on peut le voir sur la figure 3.2. Dans la deuxième partie, dont un extrait est visible en figure 3.3, les citations sont uniquement indiquées en marge, ce qui ne

donne d'indications ni sur le début, ni sur la fin de la citation. Ce changement de mise en page est attribué à la mort de H.S Horovitz au cours du travail d'édition.

פרשת בא

ויאמר יי׳ אל משה ואל אהרן בארץ מצרים לאמר. שומע אני שהיה הדבר למשה ולאהרן כשהוא אומר ויהי ביום דבר יי׳ אל משה בארץ מצרים (שמות ו כו) למשה היה הדבר ולא לאהרן. א״כ מה תלמוד לומר אל משה ואל אהרן. אלא מלמד שכשם שהיה משה כלל לדברות כך היה אהרן כלל לדברות. ומפני מה לא נדבר עמו מפני כבודו של משה. נמצאת ממעט אהרן מכל הדברות שבתורה חוץ משלשה מקומות מפני שאי אפשר. דבר אחר אל משה ואל אהרן למה נאמה לפי שהוא אומר ויאמר יי׳ אל משה ראה נתתיך אלהים לפרעה (שם ו א). אין

FIGURE 3.2 – Début du texte de la Mekhilta, Ed. Horovitz-Rabin. Les citations référencées sont en bleu, les non référencées en rouge

שנאמר ׳ועל חרבך תחיה. נגלה על בני עמון ומואב, אמר להם, מקבלים אתם את התורה. אמרו לו, מה כתוב בה, אמר להם: לא תנאף. אמרו לו, כלנו מניאוף דכתיב ׳ותהרין שתי בנות לוט מאביהם. והיאך נקבלה. נגלה על בני ישמעאל, אמר להם, מקבלין אתם עליכם את התורה. אמרו לו, מה כתוב בה, אמר להם: לא תגנוב. אמרו לו, בוו הברכה נתברך אבינו, דכתיב ׳והוא יהיה פרא אדם, וכתיב ׳כי גנב גנבתי. וכשבא אצל ישראל, ׳מימינו אש דת למו, פתחו כלם פיהם ואמרו: כל אשר דבר ה׳ נעשה ונשמע, וכן הוא אומר ׳עמד וימודד ארץ ראה ויתר גוים.

בראשית כז מ
שם יט לו
שם טז יב
שם מ טו
דברים לג ב
חבקוק ג טו

FIGURE 3.3 – Deuxième partie de la Mekhilta, Ed. Horovitz-Rabin avec les références des citations en marge

Chapitre 4

Exploration du corpus

4.1 Introduction : que recherche t'on ?

Les *midrashim* sont des commentaires de la Bible hébraïque qui, en tant que tel, contiennent un grand nombre de citations. À ce titre, la *Mekhilta* ne fait pas défaut. Pour s'en rendre compte, il suffit de regarder la densité et la diversité des types de citations qui apparaissent dans l'exemple présenté en figure 4.1. Il s'agit de la deuxième page de la *Mekhilta* d'Oxford où toutes les citations sont identifiées en rouge.

Sur cette figure 4.1 on remarque la profusion des citations, la densité variable selon les zones de la page, mais aussi la diversité des longueurs, certaines citations n'étant formées que d'un seul mot. Une analyse un peu plus fine montre aussi que beaucoup de ces citations sont introduites par l'expression שנאמר (*sheneemar*) en vert, et qu'elles sont fréquemment suivies par l'expression וגומר (*wagomer*), équivalent hébraïque de notre *etc...*, en bleu. Ces aspects révèlent d'hors et déjà quelques difficultés importantes à venir de notre travail : détecter les citations d'un seul mot, car elles ne sont pas du tout détectables avec le système de DICTA, détecter les citations qui ne sont pas introduites ou suivies par des termes fréquemment utilisés dans ce contexte et détecter l'intérieur des citations.

Dans le chapitre qui suit, nous allons explorer différents aspects de la *Mekhilta* qui vont ensuite nous aider à construire notre modèle de supervision faible à partir de méthodes heuristiques basées sur ces observations ainsi que des résultats de DICTA. Dans un premier temps, nous allons analyser les résultats de Dicta. Ensuite nous effectuerons une analyse textométrique du texte afin de repérer les récurrences statistiques nécessaires à l'élaboration de règles heuristiques.

4.2 Résultats de Dicta sur Oxford et Horovitz-Rabin

4.2.1 Oxford

Le *citation finder* développé par DICTA dispose d'une interface en ligne qui ne permet de traiter qu'un nombre limité de caractères. Pour pouvoir traiter des textes entiers, nous avons dû en faire la demande auprès de Avi Shmidman, Directeur de l'équipe DICTA. Celui-ci nous a fourni deux outputs pour la *Mekhilta* d'Oxford. Le premier a été réalisé avec un *threshold* (seuil) de 18, l'autre avec un seuil de 25. Les

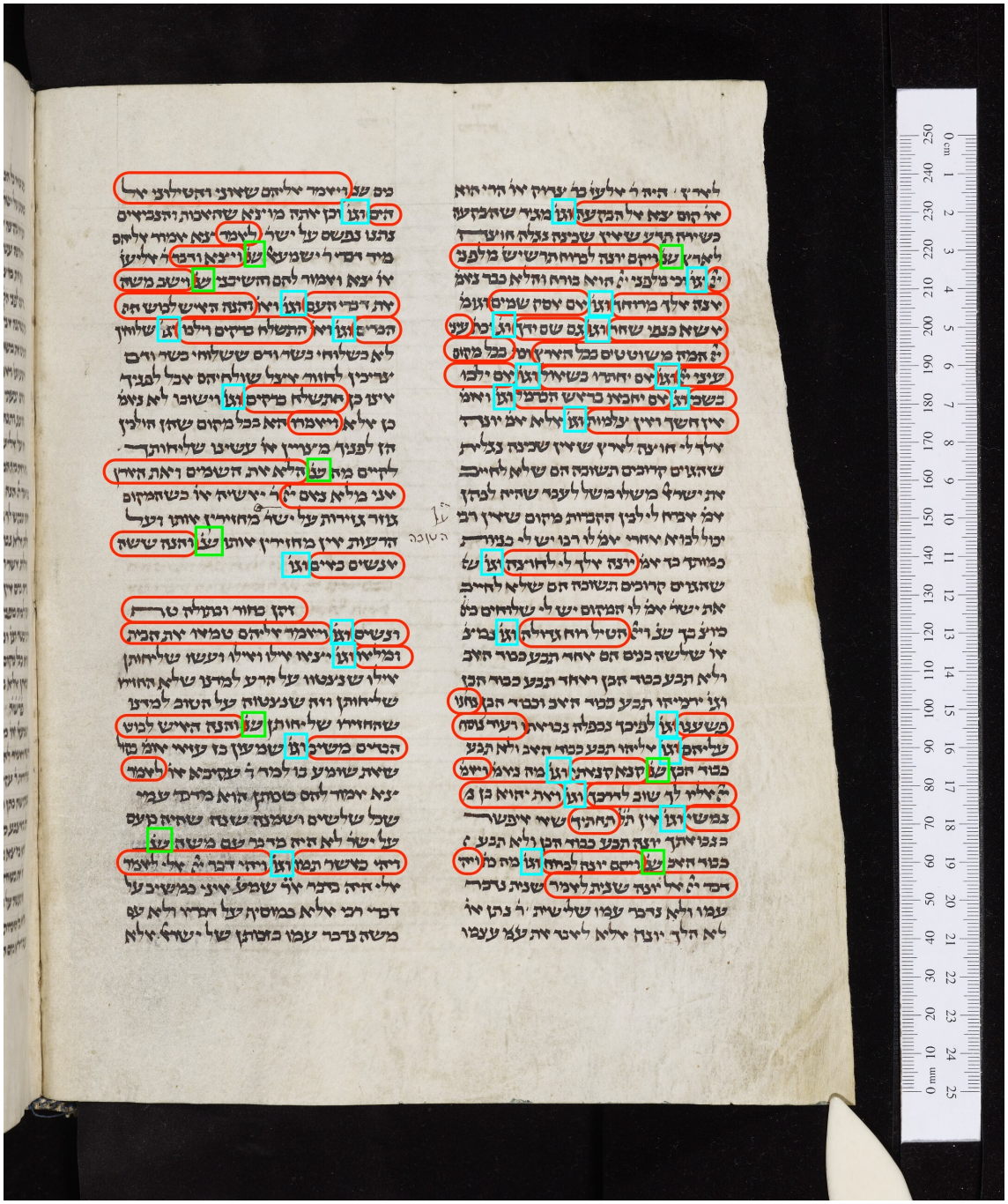

FIGURE 4.1 – Deuxième page de la *Mekhilta* - MS Oxford

résultats sont au format *json*.

Comme on peut le voir en figure 4.2, les passages sont identifiés grâce aux index du premier et du dernier caractère (champs *"startIChar"* et *"endIChar"*). Nous avons donc dû effectuer quelques manipulations pour pouvoir aligner ce résultat avec nos données (cf code en annexe). On voit aussi que le système identifie les mots considérés comme citation en utilisant la balise <b>dans le champ *"text"* et donne ensuite la ou les références possibles pour la citation repérée dans le champ *"matches"*.

```
{
  "startIChar": 328,
  "endIChar": 339,
  "text": "… חוץ משלשה מקומות  <b>יין</b> <b>ושכר</b> <b>אל</b> הנה נתתי לכם …",
  "matches": [
    {
      "mode": "Tanakh",
      "verseId": "Tanakh.Torah.Leviticus.10.9",
      "matchedText": " <b>יַיִן</b> <b>וְשֵׁכָר</b> <b>אַל</b> תֵּשְׁתְּ אַתָּה וּבָנֶיךָ",
      "score": 23.800000000000001,
      "verseiWords": [
        0,
        1,
        2
      ],
      "verseDispHeb": "ויקרא י, ט"
    }
```


FIGURE 4.2 – Output de Dicta en json

Analyse d'exemples d'erreurs et oublis d'attribution

Regardons maintenant quelles sont les citations détectées par le système. L'output avec le seuil de 25 détecte 2856 citations. Sa précision est plutôt élevée, c'est à dire qu'il est censé contenir très peu de faux positifs. Néanmoins, il manque beaucoup de citations. On remarque aussi que les citations ne sont pas toujours identifiées dans leur entièreté. Par exemple, concernant la toute première phrase de la *Mekhilta*, qui est une citation

ויאמר ייי אל משה ואל אהרן בארץ מצרים לאמר

Le système ne va pas taguer les deux premiers mots, certainement car l'expression ויאמר ייי (*wayomer adonaï*) qui signifie "Adonaï a dit" est extrêmement courante dans la Bible. Pour vérifier cette hypothèse, nous avons réalisé un test avec la citation ci-dessous, également extraite du livre de l'Exode, directement dans l'interface web :

ויאמר ייי אל משה

Adonaï a dit à Moïse. Le système ne détecte aucun mot de cette citation. Pour obtenir un résultat, il faut descendre sous le seuil de 14. En effet, dans cette citation tous les bigrams sont des bigrams extrêmement fréquents dans le *Tanakh*.

En survolant la *Mekhilta*, on peut repérer quelques cas similaires non détectés par DICTA :

הוא אהרן ומשה
ויצא ודבר

Concernant ces deux exemples, on remarque qu'ils sont précédés de l'expression introductive très fréquente dans la *Mekhilta* : תלמוד לומר (*Talmud lomar*), ce qui nous offrira un autre moyen de détecter ces citations, comme nous le verrons dans la suite de ce chapitre.

Concernant l'output avec un seuil à 18, 4395 citations sont détectées. Il y a cependant, a priori, beaucoup de faux positifs. Nous en donnons quelques exemples en fig.

4.1

Mekhilta	Bible	Verset supposé
מה לא נדבר	מה־נדבר	Genesis.44.16
יכל לבוא	ולא־יכל משה לבוא	Exodus.40.35
האב וכבוד הבן	האב וכנפש הבן	Ezekiel.18.4

TABLE 4.1 – Faux positifs dûs à des Erreurs d'attribution

Dans les trois exemples en fig. 4.1 des bigrams et trigrams de la *Mekhilta* sont considérés comme des citations par erreur du fait de la capacité du *citation finder* à prendre en compte les insertions de termes et les reformulations. En abaissant le seuil de sensibilité à 18, ce type d'erreur aura naturellement tendance à s'accroître.

Répartition des citation en fonction de la longueur

Regardons maintenant la répartition des citations détectées en fonction de la longueur

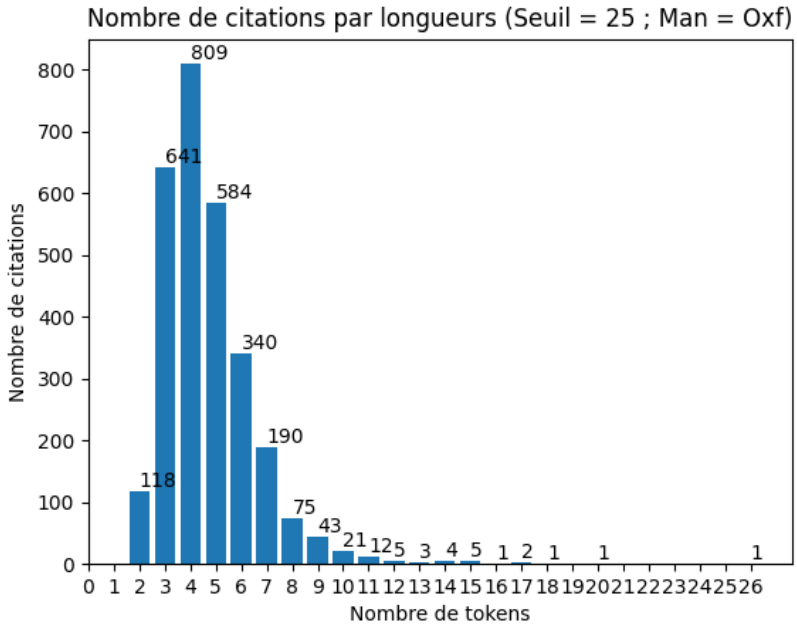

FIGURE 4.3 – DictaOxf25 : Compte par longueur de citations

Si l'on compare les graphiques 4.4 et 4.3, on remarque un bond dans la détection de citations, ou de segments de citations, de deux mots entre le seuil à 25 et le seuil à 18 et une inversion des proportions entre les citations de 3 et 4 mots. Même en considérant qu'il peut y avoir un nombre important de faux positifs parmi les citations de deux mots comptabilisées avec le seuil à 18, ce résultat indique quand même une faiblesse du seuil à 25 concernant la détection de citations de cette taille. Il est également important de noter que les citations de un seul mot sont indétectables

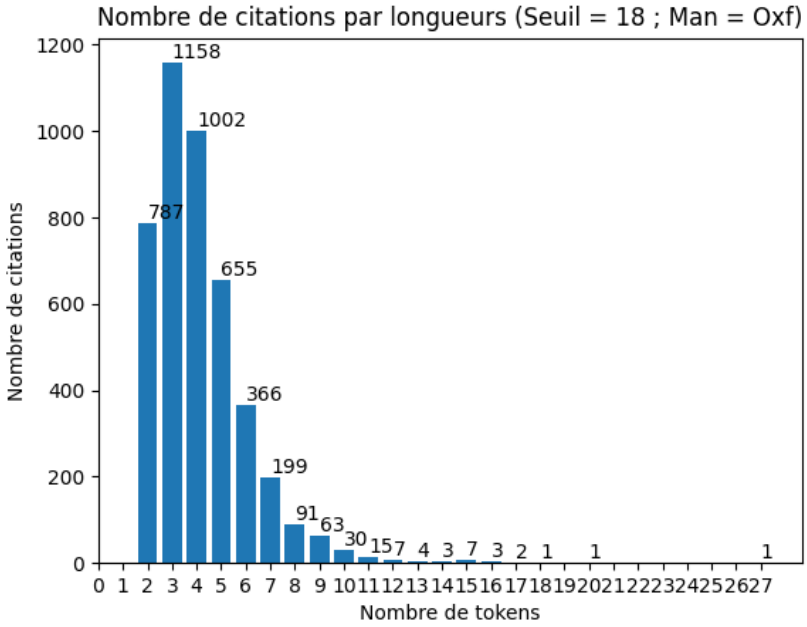

FIGURE 4.4 – DictaOxf18 : Compte par longueur des citations

pour le système, ce qui est évident au vu du fonctionnement que nous avons expliqué au chapitre 1 de la partie 1. Avi Shmidman rappel d'ailleurs [Shmidman, 2022] que la notion de réutilisation concernant un seul mot n'a pas réellement de sens. Néanmoins, il faut tout de même considérer qu'elle en a concernant la *Mekhilta*, et même la littérature rabbinique en général, car les auteurs vont parfois commenter un mot tiré d'une citation, ou même abréger une citation à un seul mot.

4.2.2 Horovitz-Rabin

Le cas de Horovitz-Rabin est différent car dans cette édition un certain nombre de citations sont référencées mais pas de manière précise. En effet, comme nous l'avons vu au chapitre 3, dans la première partie de cette édition les références sont indiquées directement dans le texte à la fin de la citation, ce qui permet d'identifier uniquement le dernier mot de la citation, tandis que dans la suite celles-ci sont indiquées uniquement en marge. En alignant les résultats de DICTA avec Horovitz-Rabin, on espère donc pouvoir identifier plus précisément les citations référencées par Horovitz-Rabin.

Le deuxième enjeu concerne la longueur des citations. Nous avons en effet constaté que dans cette édition l'expression וגומר (vagomer) apparaît beaucoup moins souvent que dans les manuscrits. Dans Oxford, וגומר apparaît à 1410 reprises, tandis que dans HR וגומר n'apparaît que 500 fois. Cela pourrait indiquer qu'une même citation pourrait avoir été rallongée dans Horovitz-Rabin. Cela semble logique étant donné que le manuscrit d'Oxford s'adressait à des érudits connaissant par coeur toutes les références bibliques, tandis que l'édition d'Horovitz et Rabin s'adresse à un public plus large. Une autre explication concerne le coût élevé du parchemin qui imposait de réduire le texte au moyen de citations plus courtes et d'un usage intensif

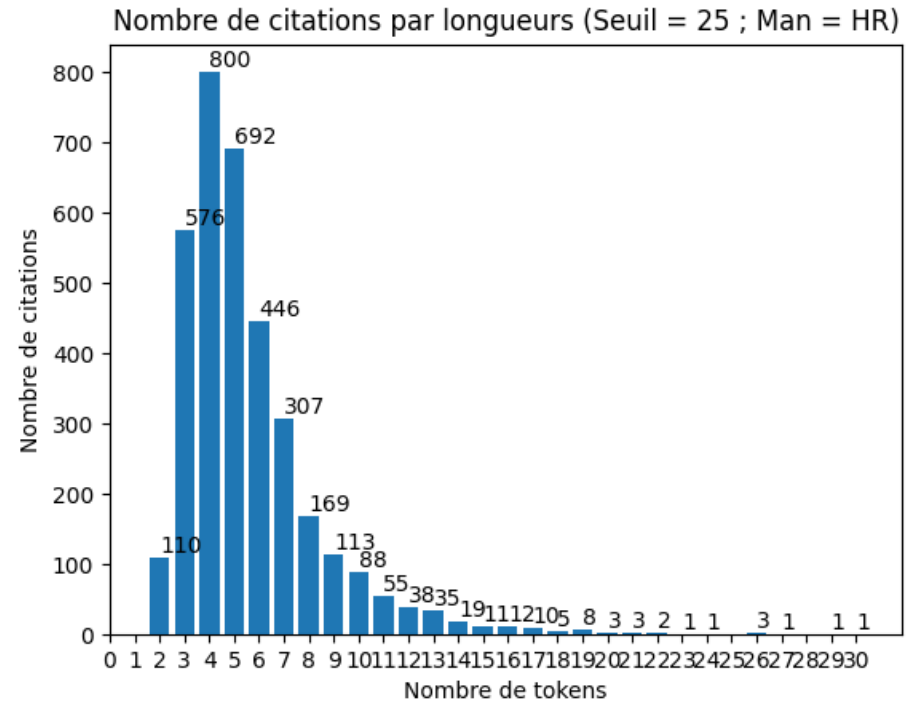

FIGURE 4.5 – DictaHR25 : Compte par longueur de citations

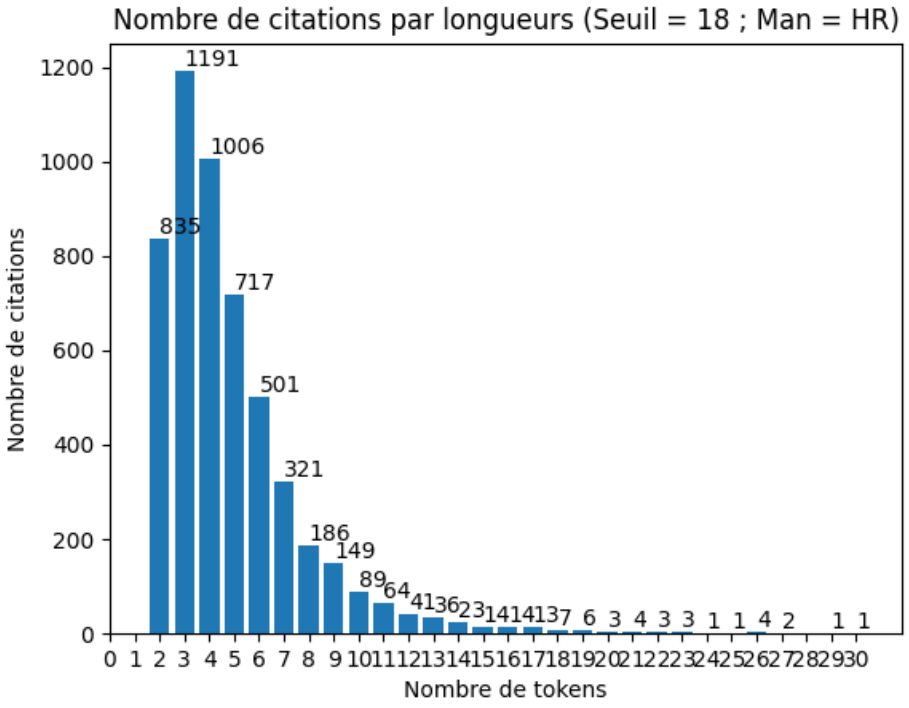

FIGURE 4.6 – DictaHR18 : Compte par longueur des citations

des abréviations. Horovitz-Rabin est une édition qui explicite beaucoup plus que les manuscrits. Nous espérons que les expériences montrerons bien que l'allongement de certaines citations, notamment celles inférieures à quatre mots, aide à la détection par DICTA.

Pour vérifier cette hypothèse de manière empirique nous avons effectué un alignement de HR avec Oxford (Oxf). L'observation empirique permet de constater l'allongement des citations dans HR. On remarque d'ailleurs que DICTA paramétré avec un seuil de 25 détecte 3510 citations dans HR, là où il n'en détectait que 2856 dans Oxf, ce qui fait une différence de 654 citations, soit 23% de plus. Cette observation pourrait faire penser que la différence de longueur des citations entre HR et Oxf facilite la détection par DICTA. Cette propriété pourrait nous être très utile car, en se servant de l'alignement entre HR et Oxf, on pourrait alors certainement détecter encore plus de citations dans Oxf. Nous vérifierons cette hypothèse dans le chapitre II en testant si l'utilisation des citations détectées par DICTA dans HR améliore la capacité de notre modèle de supervision faible à détecter les citations dans Oxf. Pour cela, nous nous servirons de l'alignement entre HR et Oxf. Nous reviendrons dans le chapitre II sur la méthodologie utilisée pour l'alignement entre manuscrits.

Comme nous l'avons fait pour Oxford, nous avons mesuré la répartition des citations en fonction de la longueur, c'est à dire du nombre de tokens détectés par DICTA. La similarité entre les courbes obtenues pour HR et Oxf est frappante. Néanmoins, le décompte par longueur dévoile d'importantes différences qui confirment la variation de 654 citations détectées par DICTA entre les deux versions de la *Mekhilta*.

Si l'on considère les *outputs* avec des seuils à 25, on observe qu'entre les longueurs 2 et 4, DICTA va détecter plus de citations dans Oxf, tandis qu'au dessus de la longueur 4, DICTA va détecter beaucoup plus de citations dans HR. Cela pourrait confirmer notre hypothèse que les citations sont plus longues dans HR. Il faut cependant rester prudent car ces longueurs correspondent au nombre de tokens détectés par citation détectée. Les citations peuvent être en réalité plus longues, ou plus courtes, que ce qui a été détecté par DICTA.

La comparaison des résultats obtenus pour HR et Oxf avec un seuil à 18 peut d'ailleurs apporter un élément de réponse. En effet, cette fois-ci les comptes vont être plus élevés pour HR, quelles que soient les longueurs détectées.

4.2.3 Conclusion sur les résultats obtenus avec DICTA

In fine, les citations les plus difficiles à détecter vont être celles qui ne sont pas trouvées par DICTA. Parmi celles-ci, nous avons déjà repéré les citations d'un seul mot et celles qui correspondent à des expressions trop fréquentes dans la Bible hébraïque, comme "Adonaï a dit" ou "à Moïse et Aaron". La méthode de Dicta, basée sur le calcul de similarité entre texte source et texte cible ne permet pas de trouver toutes les citations de manière précise. C'est à dire qu'on a le choix entre un résultat d'une grande précision mais avec un rappel faible ou un résultat avec un rappel élevé mais une précision moindre.

Pour obtenir un résultat qui favorise autant la prévision que le rappel, c'est à

dire qui augmente la proportion de vrais positifs tout en réduisant les faux positifs, une première idée pourrait-être de se baser non pas sur la structure interne de la citation, mais plutôt sur la manière dont la citation apparaît dans le texte. Autrement dit, plutôt que se baser sur la similarité, nous allons nous baser sur le contexte. C'est cette piste que nous allons explorer dans les parties suivantes avec l'étude des contextes précédent et suivant les citations (1), l'étude du code-switching (2) et l'étude de la répétition de citations (3)

D'autres méthodes consistant à combiner contexte et similarité ont été envisagées mais ne seront pas abordées ici. La première consiste à utiliser la similitude tout en restreignant le contexte de recherche. En effet, le *citation finder* de DICTA va analyser l'ensemble du *Tanakh*. Cependant, en réduisant le champ de recherche, à un livre ou même un chapitre de la Bible, certains bigrams vont alors être beaucoup moins fréquents et pourront éventuellement être plus facilement identifiables. L'idée ici va être d'utiliser la structure de la *Mekhilta*, qui est un commentaire linéaire du livre de *l'Exode*, pour circonscrire la recherche aux passages concernés par l'analyse en cours. La seconde méthode serait de chercher un moyen de débruiter les résultats de Dicta avec un seuil bas, i.e. qui ont une précision faible et un rappel plus élevé.

4.3 Recherche de patterns

La première piste que nous avons exploré est celle concernant les expressions précédant et suivant les citations. Comme le souligne Avi Shmidman [Shmidman, 2022], un certain nombre de citations sont censées être plutôt faciles à identifier car précédées par des expressions introductives très courantes dans la littérature rabbinique. Les citations étant très fréquentes, on peut supposer que ces d'expressions le seront aussi.

Le graphique 4.7 fait apparaître plusieurs tokens correspondant à ce type d'expression. תלמוד לומר est une expression très fréquente dans la littérature tannaïtique, qu'on pourrait traduire par "d'après les enseignements de la Bible". Cette expression apparaît en violet en bas du graphique 4.7 avec les deux tokens ת' et ל'. שנאמר, qui apparaît ici sous sa forme abrégée שן, est aussi une expression introductive très courante. Elle pourrait être traduite par "qui dit". L'analyse du contexte d'apparition de ces deux expressions montre qu'elles apparaissent quasi exclusivement dans le contexte de citations. Ce n'est pas forcement le cas avec d'autres tokens fréquents dans ce graphique. או' (en bleu ciel), est le token le plus fréquent de la *Mekhilta*. Il est la forme abrégée de אומר et correspond au participe présent masculin singulier du verbe dire. Cependant, selon les contextes, או' pourra introduire, ou non, une citation.

Les autres tokens représentés dans le graphique 4.7 correspondent à des mots-outils de la langue hébraïque : את, אלא, ou à d'autres mots fréquents de la langue hébraïque mais qui ne sont pas des expressions introductives : לא, qui veut dire 'non', ou ר' qui est l'abréviation de 'Rabbi'.

Pour affiner notre recherche, nous nous sommes intéressé à des ngrams de différentes longueurs, comme on peut le voir dans les graphiques 4.8, 4.9 et 4.10. L'analyse manuelle du contexte d'apparition de ces ngrams dans Oxf, nous a permis

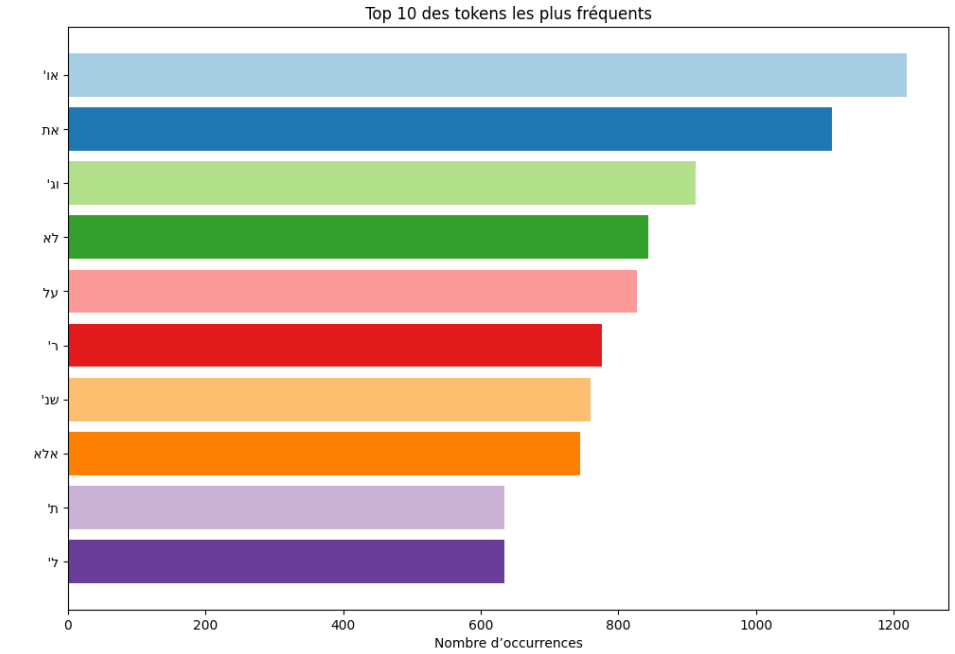

FIGURE 4.7 – Les tokens les plus fréquent dans Oxf

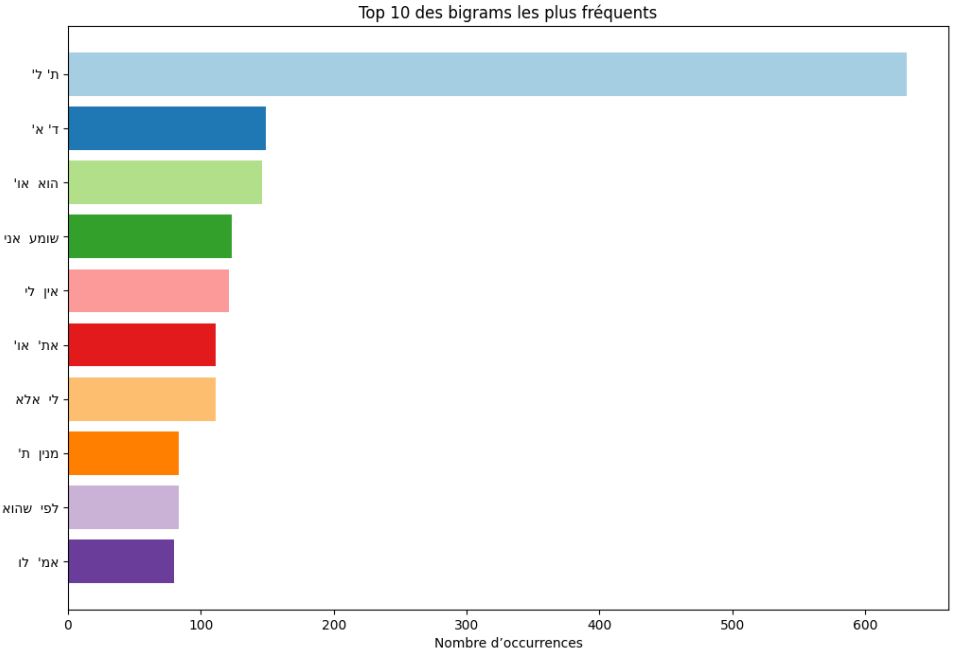

FIGURE 4.8 – Les bigrams les plus fréquent dans Oxf

de déterminer que certains précédaient exclusivement des citations, tandis que

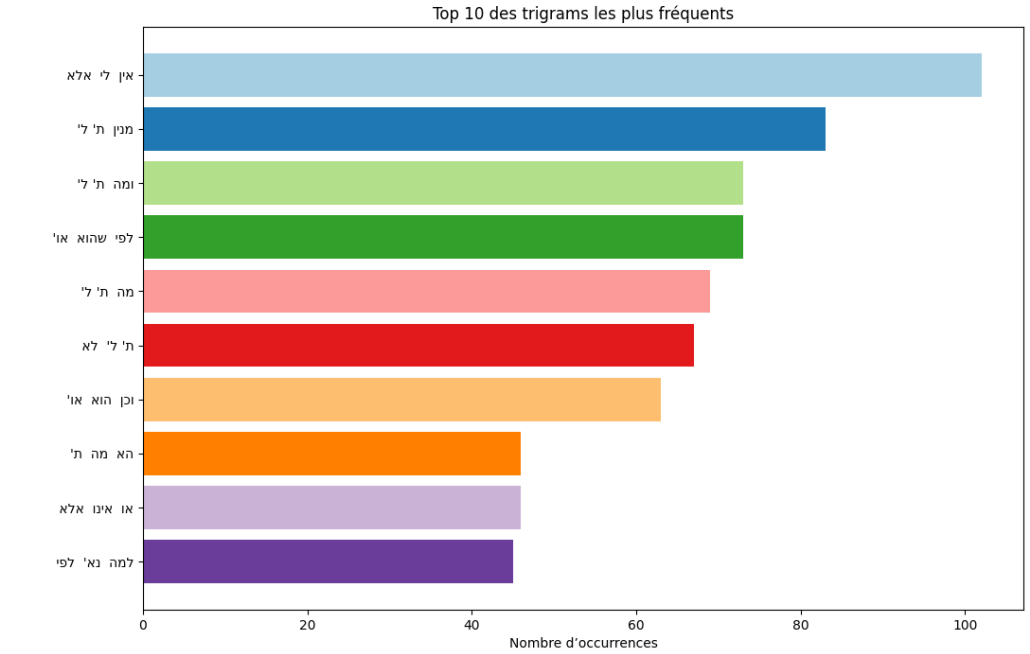

FIGURE 4.9 – Les trigrams les plus fréquent dans Oxf

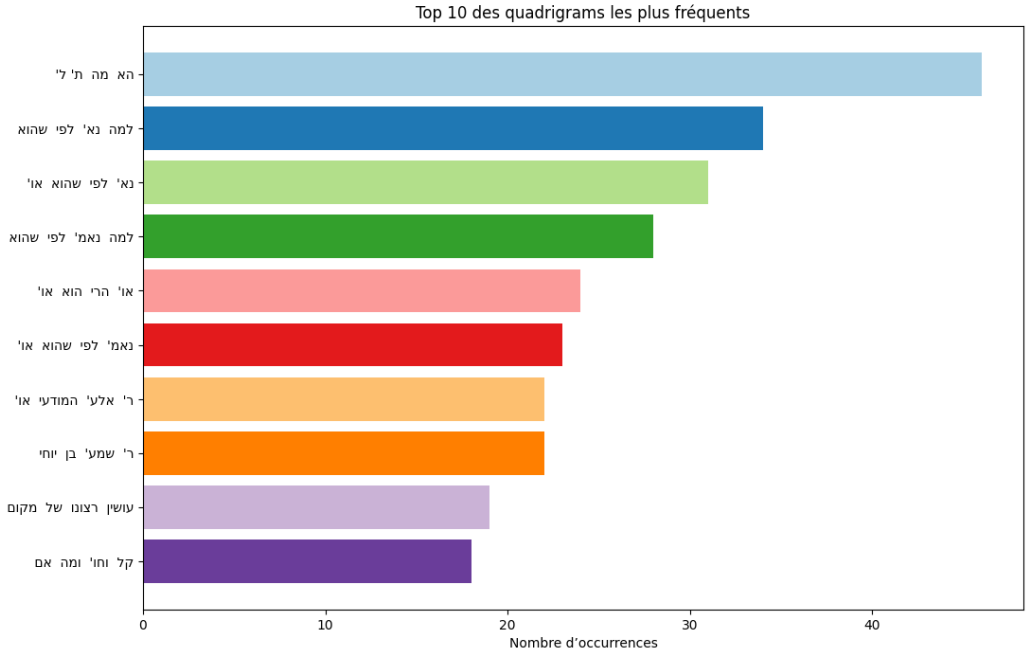

FIGURE 4.10 – Les quadrigrams les plus fréquent dans Oxf

d'autres pouvaient apparaître dans des contextes différents.

C'est le cas du token או' qui apparaît fréquemment avant les citations, mais qui peut aussi se manifester dans d'autres contextes. L'analyse des bigrams, trigrams et quadrigrams fait ressortir plusieurs expressions qui contiennent le mot או' et qui, cette fois-ci, sont spécifiques aux citations :

וכן הוא אומר
לפי שהוא אומר

En combinant mesures textométriques et observations manuelles, nous avons pu déterminer des listes de candidats pour les tokens, les bigrams, les trigrams et les quadrigrams. Pour prendre en compte toutes les citations concernées, nous avons intégré les variations orthographiques. Pour détecter ces variations, une possibilité serait d'utiliser word2vec. Comme nous avons pu le constater, word2vec permet de faire ressortir les différentes variantes orthographiques d'un même mot puisque ces différentes variantes vont apparaître dans les mêmes contextes. Néanmoins, étant donné le nombre limité de tokens concernés, l'analyse manuelle est tout aussi rapide.

Une fois obtenues nos listes de ngrams qui précèdent fréquemment les citations, il reste a identifier le reste de la citation. Une première solution va être de chercher les expressions qui suivent le plus fréquemment les citations. Comme vu précédemment, l'expression וגומר sert très fréquemment à marquer la fin d'une citation. Dans Oxf, cette expression apparaît à 1410 reprises. Nous avons également identifié d'autres expressions très fréquentes à la suite d'une citation :

להוציה
הקיש
למה נאמר לפי
שומע אני
אין לי

Néanmoins la connaissance des expressions qui suivent et précèdent fréquemment les citations n'est pas suffisante pour identifier l'ensemble de la citation. En effet, comme nous l'avons observé dans la figure 4.1, la fréquence d'apparition des citations est très importante, et les citations sont de longueurs variables. De ce fait, utiliser uniquement des marqueurs de début et de fin de citations peut conduire à un nombre important d'erreur. Pour s'en rendre compte, il suffit de se représenter l'enchaînement de trois citations avec juste quelques mots entre chaque citations. Si la première commence avec une expression introductive connu mais ne finit pas avec une expression clôturante connue et que la deuxième ne commence pas avec une expression introductive connue mais est clôturée par une expression connue du type *vagomer*, une règle de détection basée sur les expressions ouvrantes et fermantes risque de considérer les mots qui séparent les deux citations comme des citations. On observe ce type d'exemple dans l'extrait de la *Mekhilta* visible sur la figure 4.1

Pour contourner cette difficulté, nous avons essayé d'identifier des couples d'expressions précédent / suivant fréquentes en fonction de la longueur des citations. Nous avons ciblé principalement les citations de moins de quatre tokens. Ce travail a

abouti à l'établissement de listes de tokens précédant/suivant que nous avons inclus dans nos méthodes heuristiques.

4.4 L'étude du *code-switching* entre hébreu rabbinique et hébreu biblique : questions de morphologie et de syntaxe

Une autre manière d'analyser le problème de la détection de citations est de le considérer comme un problème de *code-switching*. En effet, les citations que nous cherchons à détecter sont en hébreu biblique, tandis que le reste du texte est en hébreu mishnique. Plus de 1000 ans séparent la rédaction du livre de L'Exode de la rédaction de la *Mekhilta*, faisant de l'hébreu biblique et de l'hébreu mishnique deux types d'hébreux identifiables.

Voici quelques-unes des caractéristiques qui différencient l'hébreu mishnique de l'hébreu biblique :

— Disparition de l'usage du ו pour changer l'aspect d'un verbe (*wav* conversif)
— L'infinitif est toujours introduit par un ל
— Absence d'infinitif absolu
— Absence de la forme jussive ויהי
— Absence de la forme verbale Pual
— La forme verbale Nitpael remplace le Hitpael
— La préposition של remplace les possessifs suffixés
— Le pronom personnel אנחנו remplace le pronom personnel אנו
— Le préfixe ש remplace le pronom relatif אשר
— Le démonstratif זו remplace le démonstratif זות
— Le verbe חזר est préféré au verbe שוב pour signifier *revenir*
— La syntaxe est plus simple

Nous ne détaillerons pas ici la signification exacte de ces constructions grammaticales. Ces caractéristiques liées à des innovations de l'hébreu mishnique vont permettre d'identifier des mots ou expressions qui ne sont pas des citations. Concernant les caractéristiques propres à l'hébreu biblique, en dehors des conjugaisons utilisant le wav conversif, ainsi que quelques mots-outils tels que אשר, nous n'avons pas trouvé beaucoup de caractéristiques clairement distinctives.

Pour nous aider dans l'étude du *code-switching*, nous avons aussi utilisé l'analyse morphologique fournie par DICTA. À partir de cette analyse, nous avons cherché à identifier des phénomènes morpho-syntaxiques propres aux citations. La recherche de formes syntaxiques spécifiques nous semblait être une piste prometteuse et n'a pourtant abouti à aucun résultat concret.

4.5 Se servir des citations déjà détectées ?

Une fois épuisées toutes les techniques précédemment exposées, reste encore un certain nombre de citations qui ne sont pas du tout détectées. L'idée nous est

donc venue de nous servir des citations détectées par les autres méthodes pour en trouver de nouvelles. Nous avons alors exploré deux voies. La première, qui rejoint la question du *code-switching*, a consisté à tenter de construire un lexique à partir des citations déjà détectées. L'idée était de considérer que le fait que ces citations soient dans une forme d'hébreu spécifique, puisse impliquer également un lexique spécifique.

Nous avons constaté qu'il était très compliqué de déterminer un vocabulaire qui serait propre à l'hébreu utilisé dans les citations. En effet, hébreu mishnique et hébreu biblique ont une grande part de leur lexique en commun, ce qui rend difficile de différencier les citations de cette manière. Pour vérifier cela, nous avons analysé tous les tokens utilisés dans les citations pour voir dans quels contextes ils apparaissaient. Au final, seuls les tokens ayant des caractéristiques grammaticales propres à l'hébreu biblique, notamment les conjugaisons utilisant le wav conversif, se sont avérés réellement efficaces pour identifier des mots de citations.

La seconde voie s'est basée sur l'idée que, dans la *Mekhilta*, les rabbins vont parfois chercher à approfondir certains aspects de citations en reprenant des passages spécifiques de ces citations. Nous avons donc essayé de repérer si dans les répétitions de mots, de bigrams et de trigrams dans une fenêtre contextuelle donnée se trouvaient des citations.

L'exemple ci-dessous montre un cas où l'auteur expose une citation plutôt longue avant d'en analyser deux extraits plus courts.

יי איש מלחמה יי שמו יי איש מלחמה שהוא נלחם במצרים יי שמו שהוא מרחם על בריותיו

Le début de cet extrait est une citation de Exode 15.3 qui est traduite par le Grand Rabbinat *L'Éternel est le maître des batailles, Éternel est son nom* :

יי איש מלחמה יי שמו

Cette citation va être décomposée en deux parties qui seront analysées séparément : יי איש מלחמה (*L'Éternel est le maître des batailles*) et יי שמו (*Éternel est son nom*). Voici une traduction de l'extrait avec en marron la citation originale et en bleu les reprises :

L'Éternel est le maître des batailles, Éternel est son nom.L'Éternel est le maître des batailles qui combattit en Égypte, Éternel est son nom, il a pitié de ses créatures

Cette voie s'est révélée relativement productive. Néanmoins, la technique ne fonctionne pas toujours parfaitement. Dans un long passage de la *Mekhilta* le commentateur répète plusieurs fois le bigram רוח קדים (vent d'est) sans qu'on puisse dire avec certitude s'il s'agit d'une citation :

ויולך יי את הים ברוח קדים עזה כל הלילה בעזה שברוחות ואי זו זו זו רוח קדים וכן אתה מוצה שלא נפרע המקום מאנשי דור המבול מאנשי סדום אלא ברוח קדים עזה שנאמר מנשמת אלוה יאבדו ומרוח קדים יכלו מנשמת אלוה יאבדו זה דור המבול ומתוח אפו יאכלו אלו אנשי סדום וכן אתה מוצא באנשי מגדל שלא נפרעהמקום מהם אלא ברוח קדים שנאמר ומשם הפיצם יי על פני כל הארץ ואין הפצה אלא רוח

קדים שנאמר ברוח קדים אפיצם וכן אתה מוצה במצרים שנאמר ויי ניהג רוח קדים בארץ וכן אתה מוצה שלא נפרע יי מעשרת הצבטים אלה ברוח קדים

Que l'on pourrait traduire par :

Et l'Éternel fit reculer la mer, toute la nuit, par un vent d'est impétueux le plus fort des vents, le vent d'est. Et on découvre ainsi que le Seigneur a exigé le châtiment des hommes de la génération du déluge et des hommes de Sodome avec un fort vent d'est *Un souffle de Dieu les fait périr, le vent de sa colère les anéantit.* : *Un souffle de Dieu les fait périr* — la génération du déluge ; *le vent de sa colère les anéantit* — la génération de Sodome. Et ainsi on découvre avec la génération de Babel, que le Seigneur a exigé le châtiment d'un fort vent d'est *et de là l'Éternel les dispersa sur toute la face de la terre.* "dispersa" dénote un vent d'est *Tel que le vent d'est, je les disperserai* et ainsi de l'Egypte *alors l'Éternel dirigea un vent d'est sur le pays* et ainsi des dix tribus sur lesquelles le Seigneur exigea le châtiment d'un vent d'est.

Lorsque le bigram *vent d'est* est en marron c'est une citation, lorsqu'il est en rouge ce n'est pas clair. Les citations complètes sont en italique. On voit bien ici que dans un même contexte un bigram peut-être compliqué à analyser, qu'il soit avant ou après la citation où il apparaît.

Nous avons analysé les répétitions de bigrams dans des fenêtres allant de 10 à 50 tokens avant et après le bigram concerné. Les répétitions détectées étant de l'ordre de 100 à 200 bigrams selon la taille du contexte, ce qui nous a permis de pouvoir en faire l'analyse manuelle. Cette analyse a révélé que la moitié des répétitions pouvaient être considérées comme des citations, une autre part ne pouvant pas être considérées comme citations ou étant ambiguës.

4.6 Conclusion sur l'exploration du corpus

L'exploration de la *Mekhilta* nous a permis d'ouvrir des pistes pour accéder à différents types de citations, mais aussi à différents aspects de ces citations (début, fin, etc...). Nous avons aussi à notre disposition des annotations imparfaites fournie par le *citation finder* de DICTA paramétré sur différents seuils de sensibilité. Toutes ces pistes sont très incomplètes indépendamment. Cependant, nous pensons que la combinaison de ces pistes dans un pipeline *Snorkel* pourrait nous aider à obtenir de meilleurs résultats. C'est ce que nous allons expérimenter dans le chapitre suivant.

Chapitre 5

Expériences avec *Snorkel*

Grâce aux informations récoltées au chapitre 4, nous allons maintenant pouvoir définir des règles d'étiquetage et les combiner avec l'annotation de DICTA. L'idée de construire un modèle robuste à partir d'informations imparfaites et bruitées est un principe de base de la supervision faible [Ratner et al., 2017]. Cependant, *Snorkel* propose différents modèles et la manière dont ces modèles fonctionnent reste en grande partie imprévisible, dépendant largement des données fournies en entrée. Dans la plupart des systèmes d'apprentissage automatiques traditionnels, tels que ceux présentés dans l'état de l'art, la robustesse du modèle dépend largement de la qualité des données fournies en entrée. Au contraire, la supervision faible propose de partir de données bruitées. Cette idée semble au premier abord surprenante car il faut bien que ces données bruitées contiennent une part de "vérité" exploitable par *Snorkel*. Notre intuition est donc que l'efficacité de *Snorkel* dépend d'un équilibre subtil entre fiabilité et bruitage des données. Cet équilibre, difficile à définir au départ, invite donc à des réajustements constants au cours du processus de construction du modèle. Nous allons dans ce chapitre mettre cette intuition à l'épreuve au travers de la réalisation de plusieurs expériences, en incorporant des informations de plus en plus bruitées dans notre pipeline *Snorkel*.

L'objectif sera de répondre à deux grandes questions. Tout d'abord, est-ce que la combinaison de règles imparfaites dans un modèle de supervision faible permet d'obtenir de meilleurs résultats que la simple somme des résultats de ces règles ? Ensuite, l'ajout de données bruitées, telles que celles issues de DICTA, améliore t'il réellement les performances du modèle.

Pour répondre à ces questions, nous ajouterons les règles progressivement afin d'observer comment le système se comporte. Nous commencerons par implémenter les règles déduites de l'exploration du corpus. Nous ajouterons ensuite les annotations issues de DICTA, que ce soit les annotations obtenues à partir du manuscrit d'Oxford ou à partir de l'édition d'Horovitz et Rabin alignée sur Oxford. Nous terminerons avec l'ajout de données très bruitées obtenues avec DICTA paramétré sur un seuil faible. À chaque étape nous évaluerons les résultats obtenus avec les différents modèles proposés par *Snorkel*. L'idée va être de comparer différentes configurations allant du moins bruité au plus bruité.

L'absence de corpus de test nous a dans un premier temps obligé à limiter notre analyse à comparer les sorties de chaque modèle selon les différentes configurations.

Cela correspond à un des usages de *Snorkel* : constituer un corpus annoté alors même qu'on ne dispose d'aucune données annotées, ni pour l'entraînement ni pour le test. Nous avons cependant obtenu en toute fin de processus un corpus Gold avec les citations identifiées par Daniel Stökl Ben Ezra, expert dans ce domaine. Cela nous a permis de pouvoir évaluer plus finement les performances de chaque modèle.

5.1 Corpus Gold

Dans cette partie nous allons voir quel est le score obtenu par le set d'annotations issu de l'analyse de DICTA sur le manuscrit d'Oxford avec un seuil de 25 et quels volumes de citations courtes sont détectées. Nous avons choisi le seuil à 25, car sur les deux seuils auxquels nous avons eu accès initialement, c'est celui-ci qui garantit a priori le meilleur équilibre précision/rappel. Les résultats sont visibles dans le tableau 5.1 et la figure 5.1. Les citations de un mot détectées sont des tokens qui se trouvent dans des bigrams détectés par DICTA. Il s'agit de faux positifs au sens de DICTA puisque comme nous l'avons vu, DICTA ne s'intéresse pas aux citations de un seul mot.

TABLE 5.1 – Rapport de classification pour DICTA_Oxf seuil à 25

Dicta seuil 25					
Label	**Précision**	**Recall**	**Score F1**	**Support**	**Accuracy**
0	0.91	0.96	0.93	57606	
2	0.84	0.68	0.75	17466	0.8952

5.2 Préparation des données

5.2.1 Format d'annotation

Afin de préparer au mieux nos expériences, nous avons commencé par définir un format d'annotation qui nous permettrait à la fois d'évaluer les performances de nos règles au regard de leur couverture, mais qui puisse aussi être exploitable par un futur modèle d'apprentissage automatique supervisé auquel nous soumettrons le corpus annoté obtenu avec *Snorkel*.

L'exploration du corpus réalisée dans le chapitre 4 a fait émerger trois éléments centraux dans la définition d'une citation : le début, la fin et la longueur, les citations d'un seul mot étant les plus difficiles à détecter. C'est pour cela que nous avons choisis cinq étiquettes pour annoter le corpus, en plus de l'étiquette ABSTAIN :

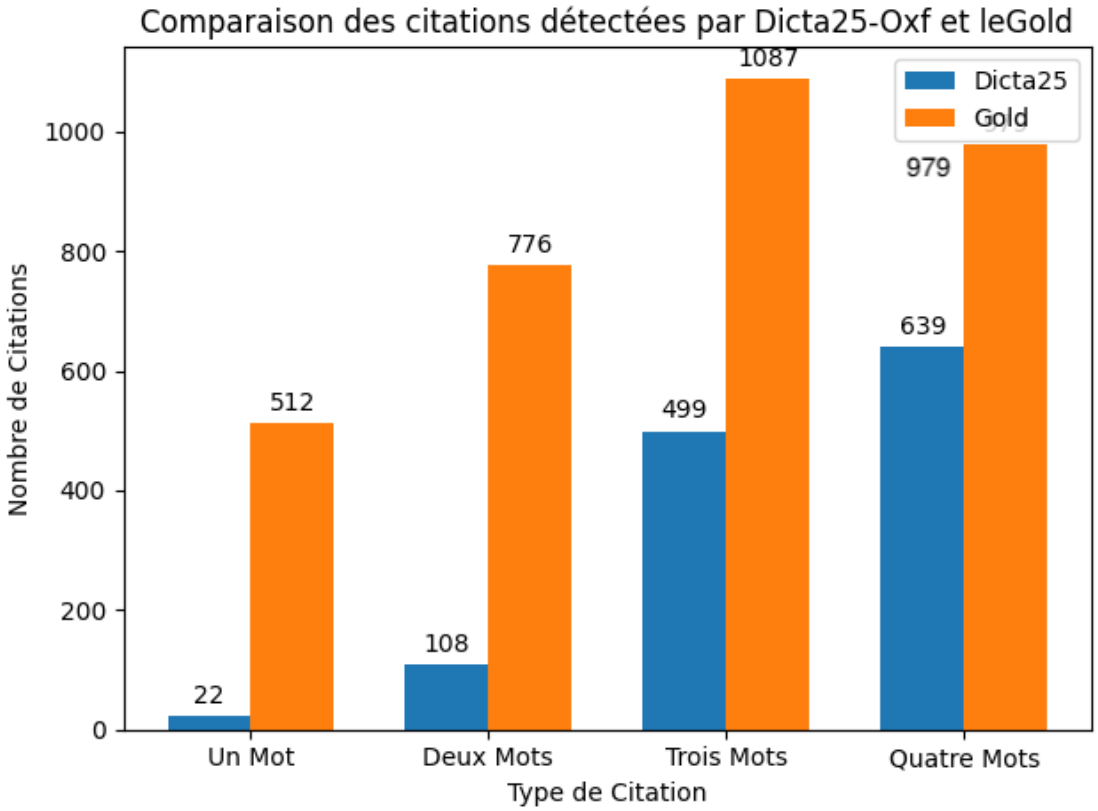

FIGURE 5.1 – Longueurs détectées par DICTA25-OXF

Étiquette	**Définition**	**Tag**
FIRST_WORD	Le premier mot de la citation	1
ONE_WORD	Citations d'un seul mot	3
LAST_WORD	Le dernier mot de la citation	4
IN_CITATION	Les mots d'une citation qui ne sont ni FIRST_WORD ni LAST_WORD ni ONE_WORD	2
NON_CIT	Tokens qui ne sont pas dans une citation	0
ABSTAIN	Tokens douteux	-1

L'étiquette ABSTAIN est le moyen pour *Snorkel* de ne pas prendre de décision sur le token en cours. Cela permet à la fois d'éviter le bruit en introduisant des prédictions incorrectes, mais aussi de mieux gérer les intéractions entre les différentes LFs. Avec ce système il est possible de spécialiser nos LFs sur certaines tâches.

Le code ci-dessous montre une LF qui va retourner LAST_WORD si c'est le dernier mot de la citation, sinon elle retourne ABSTAIN. Sans label ABSTAIN, nous aurions dû retourner NON_CIT, ce qui aurait introduit du bruit dans les données. Les modèles proposés par *Snorkel* vont tirer parti de cette étiquette pour ajuster leurs prédictions.

Listing 5.1 – LF

```
@labeling_function()
```

```
def check_next_word(x):
    return LAST_WORD if x.next_word in next_words else ABSTAIN
```

Nous verrons plus loin que ce schéma trouve ses limites avec les données issues du *citation finder* de DICTA. En effet, le *citation finder* ne détecte pas toujours les citations dans leur entièreté. Il va être alors non seulement difficile de déterminer si la longueur des citations détectées par DICTA correspondent à leur longueur réelle, mais aussi d'affirmer avec certitude si le premier et le dernier token des citations détectées par DICTA vont être réellement le premier et le dernier token de la citation.

5.2.2 Importation des données dans *Snorkel*

Snorkel intègre la capacité de traiter plusieurs types de structures de données. Parmi celles-ci, les DataFrame (DF) Pandas représentent une solution souple et pratique pour mettre en forme nos données. L'avantage de cette solution est de faciliter la transposition de nos règles en fonctions de labellisation (LFs). *Snorkel* intègre plusieurs classes qui nécessitent un DF en entrée pour fonctionner. C'est le cas de la classe `class snorkel.labeling.PandasLFApplier(lfs)` qui permet d'appliquer les LFs à des données structurées sous forme de DF.

Afin de structurer notre DataFrame de manière à ce qu'il contienne les caractéristiques qui seront utiles aux LFs, nous avons importé les colonnes correspondant aux tokens, aux identifiants et à l'analyse morphologique du manuscrit d'Oxford. Nous avons ensuite enrichi notre DataFrame en ajoutant les caractéristiques indispensables au fonctionnement de nos LFs : tokens, bigrams, trigrams et quadrigrams précédant et suivant le token en cours. Nous avons aussi ajouté des colonnes pour les tokens se situant à des distances de 2, 3, -2 et -3 du token actif.

Ensuite, nous avons importé les annotations issues du traitement du manuscrit d'Oxford par le *citation finder* de DICTA. Cela nous a permis d'ajouter deux colonnes pour les seuils à 18 et 25. Nous avons aussi ajouté les annotations de DICTA sur Horovitz-Rabin aligné avec Oxford dans une dernière colonne. En toute fin de processus, la récupération in extremis d'analyses de DICTA sur Oxford avec des seuils plus variés nous a permis d'ajouter des colonnes pour les seuils à 10, 15, 35, 45. Pour faire correspondre les annotations de DICTA avec notre DataFrame nous avons utilisé la fonction *merge* de Pandas.

5.3 Présentation des modèles proposés par *Snorkel*

Snorkel propose comme baseline le MajorityLabelVoter. Le MajorityLabelVoter est un modèle basé sur le vote majoritaire, c'est-à-dire que pour chaque datapoint, le modèle va regarder les sorties de chaque fonction d'étiquetage (LF) et prendre l'étiquette qui obtient la majorité des votes. Le rôle de l'étiquette ABSTAIN est ici très important car le MajorityLabelVoter ne comptabilise pas les abstentions. Cela signifie que si une LF retourne une étiquette supérieure ou égale à 0, et que toutes les autres s'abstiennent, le modèle choisira la LF supérieure à 0. Nous verrons plus loin que ce système peut s'avérer très robuste lorsque nous utilisons des LFs très

précises et très spécialisées.

Le modèle proposé par Snorkel est le LabelModel. Celui-ci va produire des étiquettes probables plutôt que des étiquettes déterminées. Pour cela, il va évaluer la fiabilité des étiquettes produites par les différentes LFs en se servant non seulement du consensus entre LFs mais aussi de la couverture individuelle de chaque LF, c'est à dire de la fréquence à laquelle chaque LF attribue une étiquette autre que ABSTAIN. Ces informations vont servir à attribuer des poids à chaque LF au cours du processus d'apprentissage. Le modèle va ensuite effectuer un certain nombre d'itérations pour ajuster les poids, jusqu'à converger vers une solution stable.

Dans les expériences qui vont suivre nous allons tester les performances des deux modèles conjointement, en comparant les étiquettes retournées. Nous analyserons aussi les scores obtenus avec le corpus Gold.

5.4 Implémentation des règles basées sur l'analyse du corpus

Dans cette partie, l'idée va être d'ajouter progressivement nos LFs afin de comprendre comment fonctionne *Snorkel*. L'ambition de *Snorkel* étant de combiner les LFs entre elles, il n'est pas possible d'observer son comportement avec les LFs prisent une à une. Il faut au moins trois LFs pour pouvoir tester l'outil. Néanmoins, *Snorkel* intègre un outil qui permet d'obtenir la couverture de chaque LF. Nous allons ajouter nos LFs groupées par types, avec dans un premier temps les LFs exploitant les contextes précédant et suivant, puis les couples de tokens précédant/suivant et la morphologie.

Nous comparerons les résultats obtenus avec les deux principaux modèles proposés par *Snorkel* : le LabelModel et le MajorityLabelVoter.

5.4.1 Couverture des différentes *Labelling Functions*

Afin d'avoir une idée du fonctionnement de nos deux modèles, nous commencerons par comparer les étiquettes attribuées par chaque modèle. Nous comparerons ensuite les résultats obtenus avec la couverture réelle de chaque LF. La figure 5.2 affiche la couverture de nos LFs en pourcentage et en nombre de tokens.

5.4.2 Un exemple simple

Afin de visualiser le fonctionnement des deux modèles en action, nous avons choisis de les tester avec trois LFs qui retournent chacune des labels différents. Le tableau 5.3 montre les résultats obtenus avec ces trois LFs selon le modèle, ainsi que les conflits et la couverture.

Les résultats du MajorityLabelVoter (MLV) correspondent exactement à la couverture initiale de chaque LF de laquelle on a retiré les cas conflictuels. Cela signifie que le MajorityLabelVoter a choisi de s'abstenir sur l'ensemble des cas conflictuels. Le LabelModel a, quant à lui, fait un choix sur la moitié des cas conflictuels et s'est abstenu sur l'autre moitié. Ce cas est intéressant car le MajorityLabelVoter attribue

TABLE 5.2 – Couvertures des LFs

LF	Coverage	AbsCoverage
LF_preceding_word	0.0219	1646
LF_preceding_bigram	0.0092	689
LF_preceding_trigram	0.0051	380
LF_next_word	0.0196	1468
LF_one_word_cpl	0.0011	84
LF_two_words_cpl_first	0.0022	162
LF_two_words_cpl_last	0.0022	162
LF_three_words_cpl_first	0.0057	426
LF_three_words_cpl_middle	0.0057	430
LF_three_words_cpl_last	0.0056	422
LF_morpho	0.0249	1872
LF_dicta18_INCIT	0.2571	19299
LF_dictaHR_IN_CIT	0.1949	14631
LF_dicta25_IN_CIT	0.1873	14064
LF_dicta18_NON_CIT	0.7429	55771

TABLE 5.3 – Exemple

LF	Label	Couverture	Conflits	MLV	LM
LF_preceding_word	[1]	1646	28	1618	1641
LF_next_word	[4]	1468	7	1461	1466
LF_one_word_cpl	[3]	84	25	59	61
Total	-	3198	60	3138	3168

à chaque fois l'étiquette la plus fréquemment retournée par les LFs, mais comment réagit-il lorsqu'il y a égalité des votes ?

Regardons le code source de Snorkel :

Listing 5.2 – class MajorityLabelVoter(BaseLabeler)

```
class MajorityLabelVoter(BaseLabeler):
    """Majority vote label model."""

    def predict_proba(self, L: np.ndarray) -> np.ndarray:
        """Predict probabilities using majority vote.

        Assign vote by calculating majority vote across all labeling
functions.
        In case of ties, non-integer probabilities are possible.

        Parameters
        ----------
        L
            An [n, m] matrix of labels

        Returns
        -------
        np.ndarray
```

```
        A [n, k] array of probabilistic labels

    Example
    -------
    >>> L = np.array([[0, 0, -1], [-1, 0, 1], [1, -1, 0]])
    >>> maj_voter = MajorityLabelVoter()
    >>> maj_voter.predict_proba(L)
    array([[1. , 0. ],
           [0.5, 0.5],
           [0.5, 0.5]])
    """
    n, m = L.shape
    Y_p = np.zeros((n, self.cardinality))
    for i in range(n):
        counts = np.zeros(self.cardinality)
        for j in range(m):
            if L[i, j] != -1:
                counts[L[i, j]] += 1
        Y_p[i, :] = np.where(counts == max(counts), 1, 0)
    Y_p /= Y_p.sum(axis=1).reshape(-1, 1)
    return Y_p
```

Ce code montre que la class MajorityLabelVoter(BaseLabeler) attribue, pour chaque datapoint, des probabilités à chaque étiquette. S'il y a égalité la probabilité de chaque étiquette sera équivalente. Comme la somme des probabilités doit être 1, ces probabilités seront fractionnaires, comme indiqué en commentaire : *In case of ties, non-integer probabilities are possible*. La fonction predict de la class BaseLabeler(ABC) définit ce qui arrive en cas d'égalité. D'après le code ci-dessous, 'tie_break_policy' est par défaut sur "abstain".

Listing 5.3 – class MajorityLabelVoter(BaseLabeler)

```
def predict(
        self,
        L: np.ndarray,
        return_probs: Optional[bool] = False,
        tie_break_policy: str = "abstain",
    ) -> Union[np.ndarray, Tuple[np.ndarray, np.ndarray]]:
```

5.4.3 LFs pour détecter les premiers et derniers mots de citations

Comme vu dans le chapître 4, les LFs liées à la détection du premier et du dernier mot de chaque citation ont un degré de précision très élevé. תלמוד לומר précède toujours le premier mot d'une citation, וגומר suit toujours le dernier mot. Ces LFs renvoient les étiquettes FIRSTWORD (1), LASTWORD (4) ET ABSTAIN (-1) pour étiqueter le corpus.

La figure 5.2 montre qu'une grande partie du corpus est étiquetée ABSTAIN (-1). Pour le reste, il y a presque deux fois plus de labels FIRSTWORD (1) que de labels LASTWORD (4). En utilisant ces LFs on aura donc du mal à identifier les citations dans leur entièreté. Les tokens NON-CIT ne sont pas non plus identifiés. De manière prévisible, les résultats obtenus avec uniquement les LFs permettant de détecter les premiers et derniers mots de citations sont très incomplets.

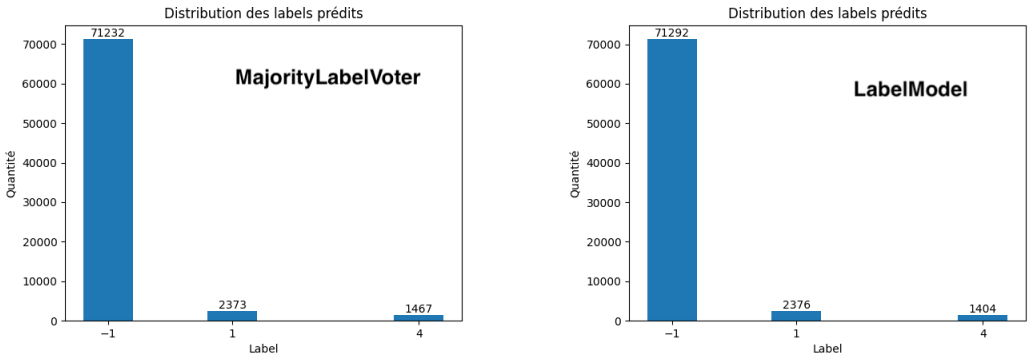

FIGURE 5.2 – Distribution des étiquettes en fonction du modèle avec les LFs preceding et next

TABLE 5.4 – Conflits entre LFS

LF	**Label**	**Couverture**	**recouvrements**	**Conflits**
LF_preceding_word	[1]	1646	45	5
LF_preceding_bigram	[1]	689	300	1
LF_preceding_trigram	[1]	380	338	0
LF_next_word	[4]	1468	8	6

Le tableau 5.4 montre les conflits entre nos quatre LFs. Si on additionne la couverture en valeur absolue des trois LFs qui retournent le label (1) et que l'on retire les 6 tokens sur lesquels il y a conflit avec la quatrième LF on trouve 2715-6= 2709 tokens. Seulement, on remarque que 2373 tokens ont été tagués, soit une différence de 336 tokens. Comment l'expliquer ? Cela est dû au fait que la couverture cumulée de nos trois LFs qui retournent le label FIRST_WORD (1) ne correspond pas à la somme des couvertures individuelles. En d'autres termes, une même donnée peut-être étiquetée par plusieurs LFs avec le même label. Dans le tableau 5.4 on voit que le nombre des recouvrements explique cette différence.

Si l'on compare les comportements des deux modèles, on constate que le MajorityLabelVoter tend à attribuer légèrement plus souvent l'étiquette LAST_WORD (4). La couverture de notre LF *check_next_word* étant de 1468 (cf. tableau 5.2), le LabelModel parait plus prudent que le MajorityLabelVoter. Concernant l'étiquette FIRST_WORD (1), on observe une différence minime dans l'attribution des deux modèles, cela est dû au faible taux de conflits.

Au final, l'analyse des résultats obtenus avec peu de LFs nous a permis d'observer des comportements légèrement différents du LabelModel et du MajorityLabelVoter. Cependant, la couverture des seules LFs associées au premier mot et au dernier mot de chaque citation est encore beaucoup trop restreinte. Nous allons maintenant introduire les classes IN_CITATION et ONE_WORD qui vont nous permettre de détecter des citations complètes.

5.4.4 Les couples précédant-suivant et morphologie

Afin d'accéder au contenu des citations, c'est à dire les tokens des citations qui ne sont ni des premiers ni des derniers mots de citations, nous allons maintenant introduire des LFs permettant de détecter les classes IN_CITATION et ONE_WORD. Pour cela nous avons effectué des encadrements à l'aide de couples de mots suivants / mots précédents fréquemment observés pour des longueurs de citations allant de 1 à 3 tokens. nous avons concentré nos recherches sur les citations de longueur 1, 2 et 3 car les citations les plus courtes sont les plus difficiles à détecter.

L'implémentation de ces couples a nécessité la création de règles spécifiques pour chaque token à annoter. Une pour le premier token de la citation, une pour le dernier et une pour le token à l'intérieur des citations de longueur 3. Le code ci-dessous présente les différentes LFs nécessaires pour annoter les citations de longueur 1, 2 et 3 avec les couples.

Listing 5.4 – Code Python pour détecter les citations de 1 à 3 mots avec les couples

```
"""
COUPLES PRECEDING/NEXT
"""

#citations de un seul mot
@labeling_function()
def check_words_before_and_one_after(x):
    for word_after, word_before in preceding_next_one:
        if x['preceding_word'] == word_before and x['next_word'] ==
            word_after:
            return ONE_WORD
    return ABSTAIN


"""
Encadrement de citations de deux mots
"""
@labeling_function()
def check_words_before_and_two_after(x):
    for word_after, word_before in preceding_next_two:
        if x['preceding_word'] == word_before and x['two_words_after']
            == word_after:
            return FIRST_WORD
    return ABSTAIN

#On cherche le dernier mot de la citation
@labeling_function()
def check_words_after_and_two_before(x):
    for word_after, word_before in preceding_next_two:
        if x['next_word'] == word_after and x['two_words_before'] ==
            word_before and x.word not in next_words:
            return LAST_WORD
    return ABSTAIN

"""
```

```
Pour les encadrements de citations de 3 mots
"""
@labeling_function()
def check_words_before_and_three_after(x):
    for word_after, word_before in preceding_next_three:
        if x['preceding_word'] == word_before and
           x['three_words_after'] == word_after and x['word'] not in
           preceding_all:
            return FIRST_WORD
    return ABSTAIN

#On cherche le dernier mot de la citation
@labeling_function()
def check_words_after_and_three_before(x):
    for word_after, word_before in preceding_next_three:
        if x['next_word'] == word_after and x['three_words_before'] ==
           word_before and x.word not in next_words:

            return LAST_WORD
    return ABSTAIN

#on cherche le mot du milieu
@labeling_function()
def check_middle_words_after_and_three_before(x):
    for word_after, word_before in preceding_next_three:
        if x['two_words_after'] == word_after and
           x['two_words_before'] == word_before:

            return IN_CITATION
    return ABSTAIN
```

La figure 5.3 montre bien l'apparition de deux nouvelles classes : IN_CITATION (2) et ONE_WORD (3). L'incertitude identifiée par l'étiquette ABSTAIN (-1) baisse légèrement par rapport aux résultats obtenus précédemment, mais surtout on voit que la quantité de FIRST_WORD (1) détectés augmente tandis qu'on perd une cinquantaine de LAST_WORD (4). Comment expliquer cette baisse dans la détection des LAST_WORD (4)?

À première vue, on peut penser que des citations de un seul mot précédemment étiquetées LAST_WORD (4) sont maintenant étiquetées ONE_WORD (3). En effet, pour des citations d'un seul mot, ce mot va être tout à la fois le premier mot et le dernier mot de la citation. De ce fait, les LFs détectant les FIRST_WORD, LAST_WORD ET ONE_WORD peuvent se contredire. Cependant, en retirant une à une les LFs de notre modèle, nous avons observé que c'est plutôt la LF liée à la morphologie qui fait baisser le volume du label LAST_WORD . Ce phénomène laisse penser qu'il y a un conflit entre la LF liée à la morphologie et les autres LFs, notamment celle qui permet de détecter les citations de un seul mot à partir des couples.

Pour vérifier cette hypothèse, Nous sommes allés chercher dans les données des cas de chevauchement. À plusieurs endroits nous avons trouvé des tokens auxquels pouvait être attribué aussi bien l'étiquette LAST_WORD avec la LF *check_next_word* que l'étiquette IN_CITATION avec la LF *pre_waw_main_VERB*. Voici un exemple :

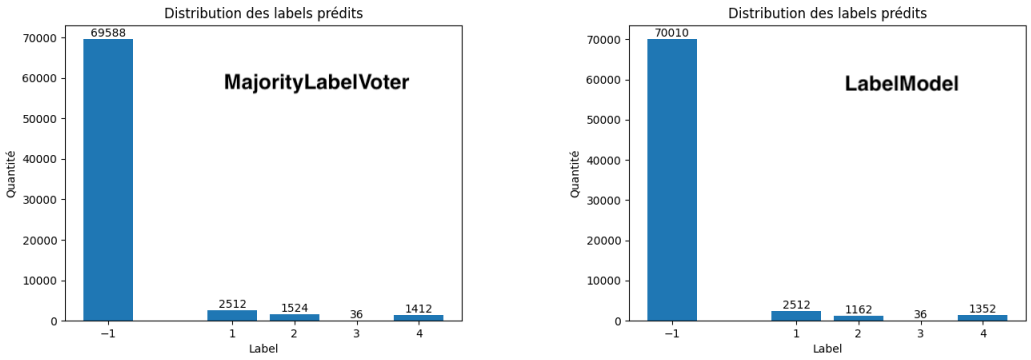

FIGURE 5.3 – Distribution des étiquettes en fonction du modèle avec les LFs preceding-next, couples et morphologie

הוא אומר משגיא לגוים ויאבדם וגומר

Dans cette exemple, le token ויאבדם est un verbe conjugué avec le wav conversif, mais il est aussi le dernier mot de la citation, suivi de l'expression וגומר.

On observe aussi que la couverture de la LF *check_next_word* ne correspond pas tout à fait au nombre de LAST_WORD détectés par le LabelModel. La LF *check_next_word* a une couverture de 1468 tokens, là où 1352 tokens sont étiquetés LAST_WORD par le LabelModel, et 1412 par le MajorityLabelVoter. Encore une fois, du fait des conflits et des recouvrements, les deux modèles vont détecter un peu moins que ce qui aurait pu être déduit de la couverture réelle de nos LFs, le MajorityLabelVoter restant plus fidèle aux informations de couverture que le LabelModel.

5.4.5 Résultats partiels

À ce stade nous n'avons ajouté que des LFs dont nous connaissions à l'avance la précision élevée. La majeure partie du corpus reste annotée ABSTAIN (-1), laissant encore une grande part d'incertitude dans les données annotées par *Snorkel*. il nous faut maintenant trouver un moyen de réduire l'abstention en introduisant des LFs qui vont étiqueter la partie du corpus dont les tokens ne font pas partie de citations. C'est ce que nous allons faire dans la partie suivante, en alimentant nos deux modèles avec les annotations issues de DICTA

Nos annotations précédentes étant très précises mais avec une couverture trop faible du corpus, nous allons maintenant ajouter des annotations moins précises mais avec une plus grande couverture du corpus. Nous allons donc observer comment réagissent les modèles à l'ajout d'informations bruitées. Dans cette partie, nous introduirons l'utilisation de la vérité de terrain fournie par Daniel Stökl Ben Ezra pour évaluer les performances de nos deux modèles selon les configurations expérimentées.

5.5 Ajout et analyse des annotateurs externes

5.5.1 Annotations fiables

Comme nous l'avons vu dans le chapitre 4, nous avons récupéré deux sets d'annotations du manuscrit d'Oxford réalisés par le *Citation Finder* de DICTA. Le set avec un seuil à 25 est très précis mais avec un rappel faible, c'est à dire qu'il couvre une faible partie du corpus en termes de citations détectées. L'output avec un seuil à 18 a un rappel plus élevé mais une précision plus faible, c'est à dire qu'il contient beaucoup de faux positifs.

À partir de ces deux sets, nous avons exploité l'idée que les résultats avec un seuil élevé vont avoir une grande précision dans la détection des citations mais une faible précision dans la détection des non-citations, tandis que les résultats avec un seuil plus bas vont avoir une faible précision dans la détection des citations mais une plus grande précision dans la détection des non-citations. En effet, le *Citation Finder* paramétré avec un seuil plus bas va détecter beaucoup plus de citations car il va trouver beaucoup plus de bigrams communs entre la *Mekhilta* et la Bible hébraïque. On peut donc supposer que les tokens non détectés par le *Citation Finder* ont plus de chances d'être des non-citations. Ces deux informations vont nous permettre d'introduire le label NON_CIT dans nos LFs et de réduire la quantité d'étiquettes ABSTAIN obtenues avec les LFs précédentes

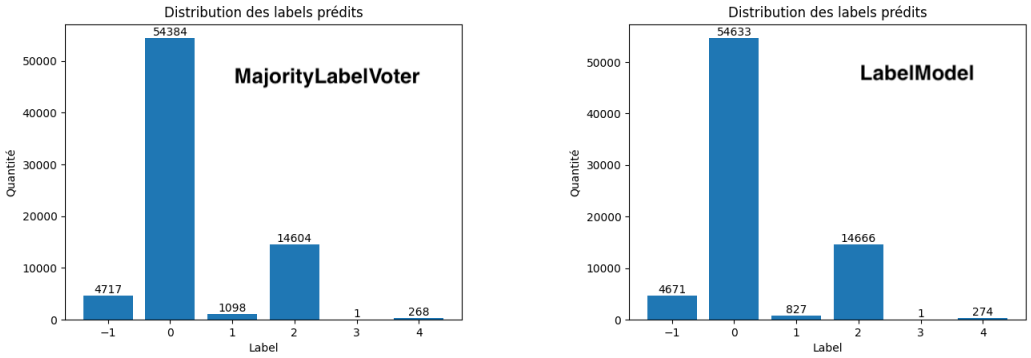

FIGURE 5.4 – Distribution des étiquettes en fonction du modèle avec les LFs preceding-next, couples, morphologie et DICTA_Oxf th18 et th25

Les annotations produites par DICTA ne sont pas parfaites. Du fait qu'elles se basent sur la fréquence d'apparition des ngrams du texte cible dans le texte source, un certain nombre de citations ne vont pas être détectées car trop fréquentes dans le corpus source. La conséquence est que certaines citations ne vont être que partiellement détectées, les groupes de tokens dont la fréquence ne permet pas de dire s'il s'agit d'une citation ou non étant considérés comme des non-citations. La conséquence de cela va être qu'on ne va pas pouvoir déterminer si tel ou tel token est le premier ou le dernier de la citation. C'est seulement en association avec nos autres règles que *Snorkel* va pouvoir déterminer si tel ou tel token est bien le premier ou le dernier mot de la citation. De ce fait, nous avons choisis d'annoter IN_CITATION les tokens détectés comme citations par DICTA, charge à *Snorkel* de se servir de nos précédentes LFs pour affiner les étiquettes.

Néanmoins, comme le montre la figure 5.4, les étiquettes FIRST_WORD (1), LAST_WORD (4) et ONE_WORD (3) sont beaucoup moins attribuées que précédemment. On peut supposer que cela est dû à la couverture beaucoup plus importante des annotations apportées par les analyses de DICTA. Nous avons vu précédemment que le taux de couverture des différentes LFs est l'un des facteur dont se servent les modèles pour évaluer les prévisions de nos LFs, ce qui pourrait expliquer cet écrasement des FIRST_WORD (1), LAST_WORD (4) et ONE_WORD (3).

Dans le rapport de classification visible dans le tableau 5.5 le score global d'accuracy est honorable mais très inférieur au score obtenu par DICTA_Oxf 25 seul. Ce score est très proche entre les deux modèles, avec une légère avance pour le MajorityLabelVoter. Il masque néanmoins de grandes disparités entre classes, ce que confirment les faibles scores obtenus pour les étiquettes FIRST_WORD (1) et LAST_WORD (4). L'étiquette ONE_WORD (3) obtient même un score à 0 sur toutes les métriques étant donné que les deux modèles n'ont assigné ce label qu'à un seul token.

TABLE 5.5 – Rapports de classification avec les annotations issues de DICTA sur Oxford (th18 et th25)

MajorityLabelVoter avec DICTA_Oxf 18 et 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.95	0.90	0.92	57606	0.792
1	0.82	0.17	0.29	4136	
2	0.55	0.80	0.65	8682	
3	0.00	0.00	0.00	512	
4	0.84	0.03	0.06	4136	
LabelModel avec DICTA_Oxf 18 et 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.95	0.90	0.92	57606	0.7907
1	0.38	0.10	0.16	4136	
2	0.49	0.82	0.61	8682	
3	1.00	0.00	0.00	512	
4	0.72	0.05	0.09	4136	

Face à ce problème, nous avons choisis de réduire l'espace des étiquettes à 3 étiquettes : ABSTAIN (-1), IN_CITATION (2) ET NON_CIT (0). Le tableau 5.6 montre les résultats obtenus.

Le score global est bien meilleur qu'avec 6 classes, même s'il masque les performances du modèle concernant les début et fin de citations. Ces scores ne dévoilent pas non plus les scores obtenus par longueurs de citations, très importants pour nous car, comme nous l'avons vu dans le chapitre 4, le *citation finder* de DICTA ne permet pas de détecter les citations de un seul mot.

Les résultats exposés au tableau 5.6 montrent des résultats assez similaires entre les deux modèles concernant la classe NON_CIT (0). Pour la classe 2 (IN_CITATION) le MajorityLabelVoter a une bien meilleure précision, mais au prix d'un rappel

TABLE 5.6 – Rapports de classification avec trois classes

MajorityLabelVoter avec DICTA_Oxf 18 et 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.95	0.88	0.92	57606	
2	0.84	0.73	0.78	17466	0.848
LabelModel avec DICTA_Oxf 18 et 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.95	0.89	0.92	57606	
2	0.79	0.76	0.77	17466	0.8558

moins élevé, tandis que le LabelModel détecte plus de citations mais avec moins de précision. Ces scores tendent à montrer un plus grand équilibre dans les résultats obtenus avec le LabelModel, ce qui lui permet d'obtenir un score d'accuracy légèrement plus élevé que le MajorityLabelVoter. Globalement les performances des deux modèles sont assez similaires.

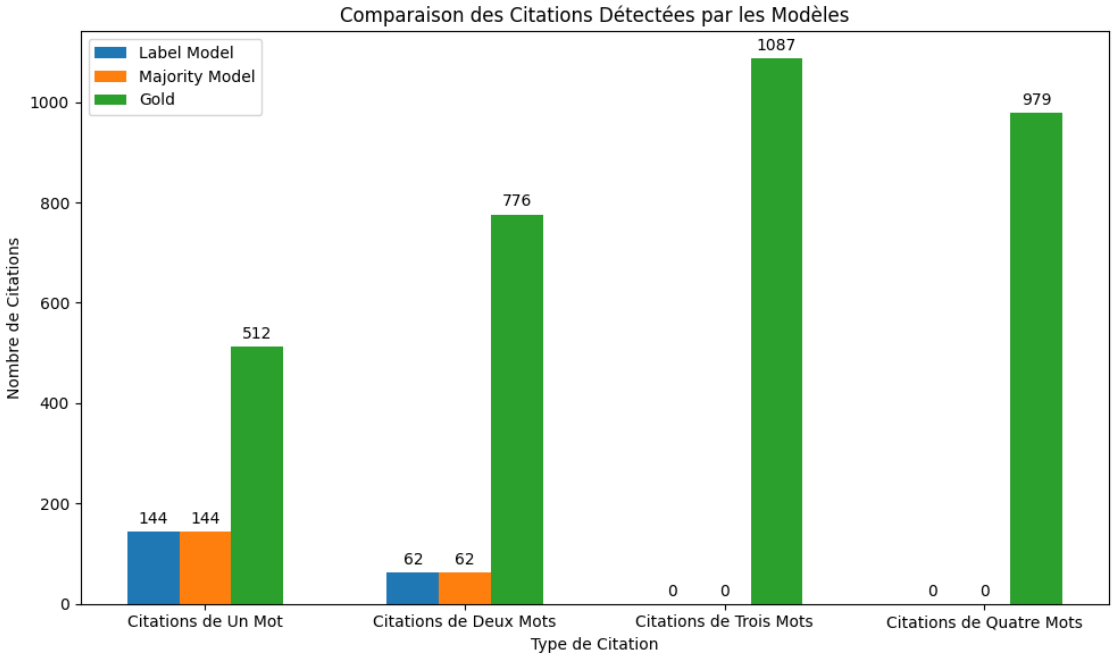

FIGURE 5.5 – Longueurs détectées avec les ngrams précédants et suivants

Les figures 5.5, 5.6 et 5.7 montrent l'évolution dans la détection des citations courtes entre les configurations avec uniquement des règles basées sur l'analyse du corpus (5.5, 5.6) puis avec l'ajout des annotations de non-citations et de citations de DICTA_Oxf avec des seuils à 18 et 25. On remarque que le nombre de citations de un seul mot baisse lorsqu'on intègre DICTA, surtout pour le MajorityLabelVoter, alors que les règles pour détecter ces citations sont toujours actives. Les annotations de

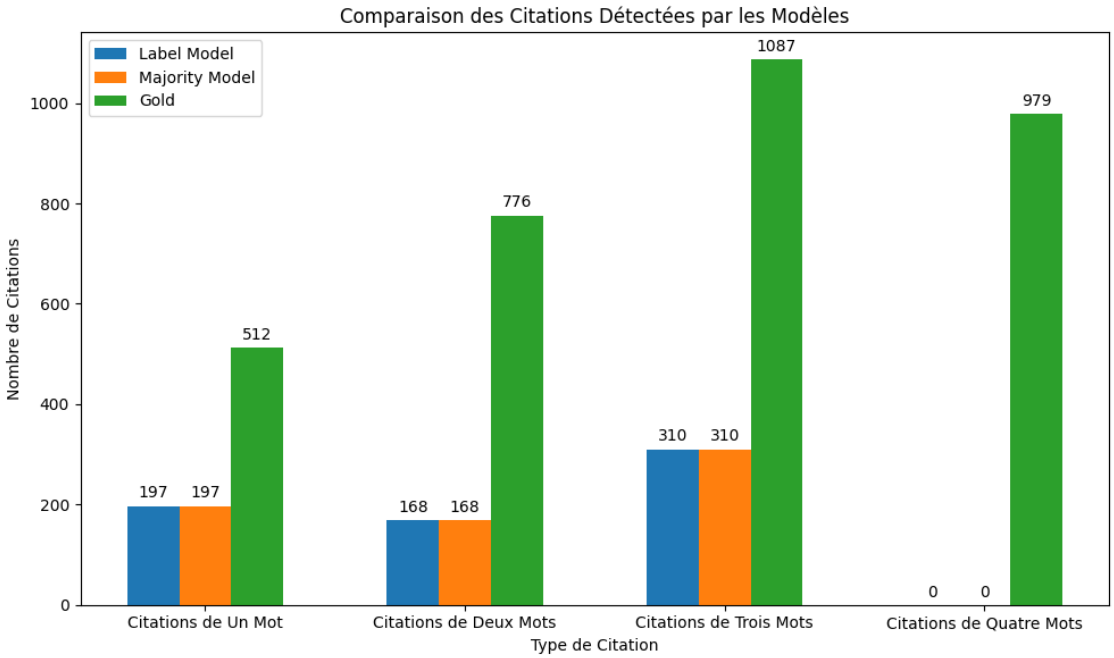

FIGURE 5.6 – Longueurs détectées avec les couples

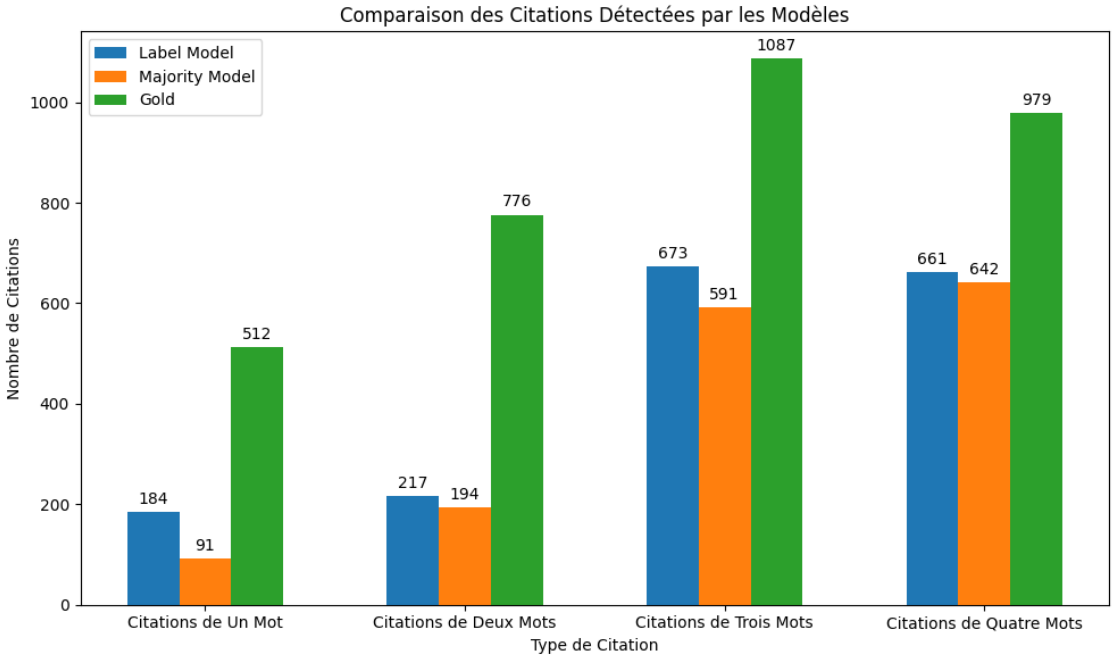

FIGURE 5.7 – Longueurs détectées avec dicta 18(NON_CIT) et dicta25(CIT)

DICTA apportent une amélioration significative dans la détection des citations de 2 à 4 mots, le LabelModel en détectant plus.

5.5.2 Ajout des annotations issues de l'édition Horovitz-Rabin

Nous allons maintenant analyser les résultats obtenus en incorporant les annotations issues de l'output du *citation finder* sur l'édition de Horovitz et Rabin. Le tableau 5.7 montre les scores de ces annotations après alignement sur Oxford.

TABLE 5.7 – Rapports de classification pour DICTA_HR seuil à 25

Dicta seuil 25					
Label	**Précision**	**Recall**	**Score F1**	**Support**	**Accuracy**
0	0.92	0.97	0.94	57606	
2	0.87	0.73	0.80	17466	0.9133

L'idée que nous allons exploiter ici est que les citations étant plus longues dans l'édition de Horovitz-Rabin que dans les manuscrits, on suppose que DICTA va en détecter plus, ce que nous avons vérifié dans le chapitre 4. Nous espérons aussi que la quantité de citations courtes détectées sera plus importante. Pour obtenir ces annotations et les incorporer à notre pipeline *Snorkel* nous avons tout d'abord dû réaliser un alignement de Horovitz-Rabin avec Oxford afin de pouvoir reporter les étiquettes issues d'Horovitz-Rabin sur Oxford.

Algorithme de Smith-Waterman

Pour réaliser cet alignement nous nous sommes servis de l'algorithme de Smith-Waterman dont l'implémentation en Python se trouve en annexe. Smith-Waterman est un algorithme initialement destiné à l'alignement de séquences d'ADN, mais qui peut aussi s'avérer très efficace pour l'alignement de séquences de mots. Le principe est assez simple, il va s'agir d'introduire des espaces, appelés *indel* en biologie, dans l'une et l'autre des séquences à aligner pour maximiser le nombre de correspondances. Dans notre cas, l'algorithme est un peu simplifié car il ne tient pas compte des similitudes qui pourraient exister entre certains tokens.

Pour réaliser l'alignement, l'algorithme va commencer par regarder chaque paire de mots et retourner un score selon trois configurations : correspondance, suppression et insertion. Les scores les plus élevés pour chaque paire vont être pondérés en fonction de valeurs de coût prédéfinies pour les correspondances, non-correspondances et les écarts (*match_score*, *mismatch_cost* et *gap_cost*) puis entrés dans une matrice. Enfin, une opération de *backtracking* va permettre de reconstruire l'alignement optimal en retraçant le chemin des décisions prises pour chaque paire de mots. La figure 5.8 montre un exemple étape par étape avec deux séquences simples.

Dans le cas de l'alignement entre l'édition d'Horovitz et Rabin et le manuscrit d'Oxford, Smith-Waterman nous a permis d'obtenir un premier résultat qu'il a été nécessaire de corriger ensuite. Nous avons pu automatiser la correction de certaines erreurs récurrentes à l'aide de Python. Le reste des corrections a dû être réalisé manuellement.

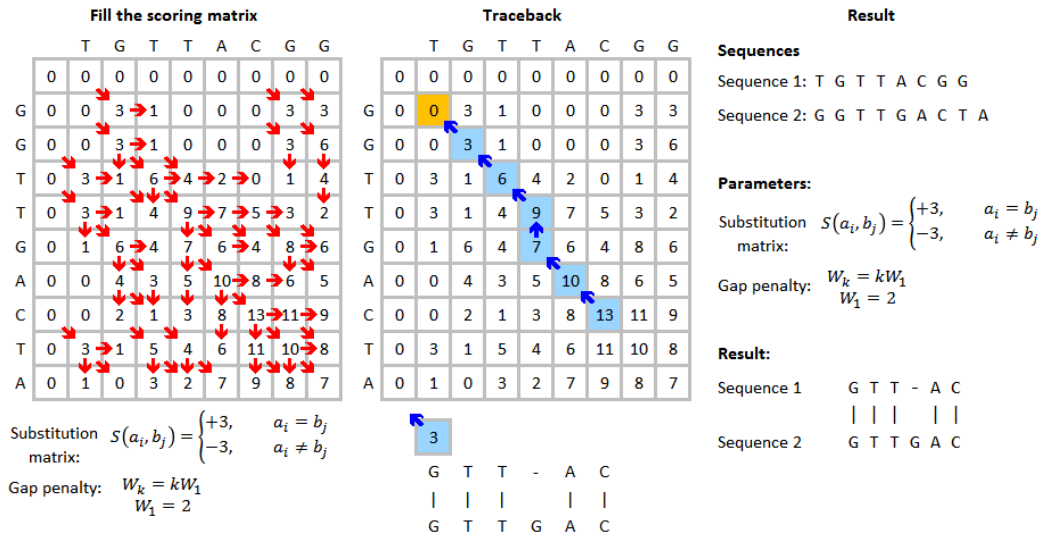

FIGURE 5.8 – Smith-Waterman : Matrice, Backtracking et Résultats -Wikipedia user Yz cs5160 - CC BY-SA 4.0

Résultats

Pour cette expérience, nous n'avons importé que les étiquettes des citations, la LF retournant ABSTAIN pour le reste. Le tableau 5.8 montre les scores obtenus. Par rapport aux résultats précédents, on observe une amélioration de la précision du LabelModel pour la classe 2, mais aussi du MajorityLabelVoter sur la classe 0. Au niveau des performances globales, c'est le LabelModel qui bénéficie le plus de ces annotations apportées par Horovitz-Rabin. Son rappel est meilleur que le rappel de DICTA_HR 25 ou de DICTA_Oxf 25 seuls, mais le score d'accuracy global reste inférieur aux scores de ces deux annotateurs seuls.

TABLE 5.8 – Rapports de classification pour MajorityLabelVoter et LabelModel avec annotations HR

MajorityLabelVoter avec annotations DICTA_HR 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.97	0.87	0.92	57606	
2	0.83	0.78	0.80	17466	0.8498
LabelModel avec annotations DICTA_HR 25					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.96	0.89	0.93	57606	
2	0.84	0.77	0.80	17466	0.8623

La figure 5.9 montre une nette progression dans la détection des citations de 3 et 4 mots, mais une baisse pour les citations de un seul mot avec le LabelModel.

5.5.3 Ajout d'annotations de Dicta avec un seuil à 18

Nous allons maintenant voir comment nos deux modèles réagissent à l'ajout d'annotations très imparfaites issues du set d'annotations de DICTA sur Oxford avec

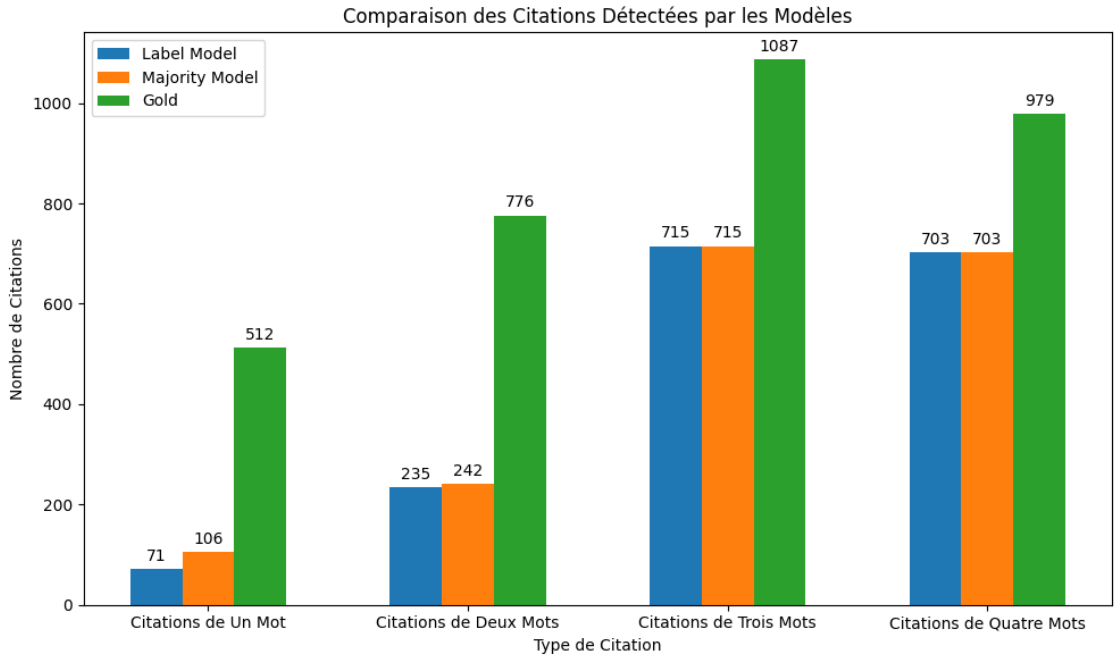

FIGURE 5.9 – Longueurs détectées avec oxf18(NON_CIT), oxf25(CIT) et HR25(cit)

un seuil à 18. Pour cela, nous avons créé une LF qui renvoie IN_CITATION lorsque le token est détecté par DICTA comme citation, ABSTAIN pour le reste.

Le tableau 5.9 montre le comportement de nos deux modèles lorsqu'on ajoute ces annotations. Le score d'accuracy montre une augmentation des performances globales de nos deux modèles, avec de meilleures performances pour le LabelModel. Au niveau des classes, on observe une stagnation des performances pour la classe 0 (NON_CIT) avec les deux modèles. Concernant la classe 2 (IN_CITATION), les deux modèles vont détecter plus de citations mais avec une moindre précision. L'ajout de données moins précises permet donc d'améliorer les performances globales de nos deux modèles, mais cela se fait au prix d'une moindre précision pour la classe IN_CITATION. Encore une fois ici, c'est le LabelModel qui l'emporte sur le score d'accuracy global, mais le MajorityLabelVoter trouve plus de citations pour une précision équivalente sur cette classe.

TABLE 5.9 – Rapports de classification pour MajorityLabelVoter et LabelModel avec IN_CITATION de DICTA_Oxf 18

MajorityLabelVoter avec IN_CITATION de DICTA_Oxf 18					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.97	0.87	0.92	57606	
2	0.72	0.83	0.77	17466	0.8614
LabelModel avec IN_CITATION de DICTA_Oxf 18					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.96	0.89	0.93	57606	
2	0.72	0.80	0.76	17466	0.8705

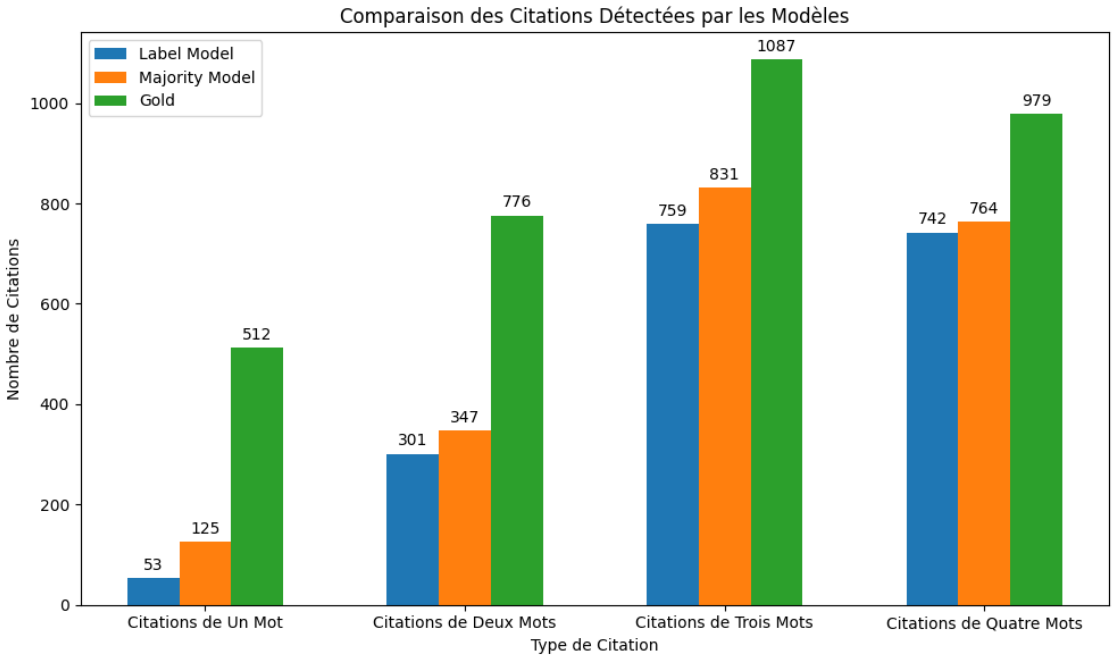

FIGURE 5.10 – Longueurs détectées avec oxf18(NON_CIT), oxf25(CIT), HR25(cit) et oxf18(CIT)

La figure 5.10 montre pour la première fois un avantage du MajorityLabelVoter dans la détection des citations courtes, avec un net progrès pour la détection des citations de 2 à 4 mots. Comme l'indique le tableau 5.9, cette progression est au prix d'une diminution de la précision dans la détection des citations (classe 2).

5.5.4 En intégrant les NON_CIT de DICTA pour différents seuils

Pour les expériences qui vont suivre nous avons changé de perspective. Il ne s'agit plus de se servir des seuils les plus élevés pour détecter les citations et des seuils les plus faibles pour trouver les non-citations. Nous allons plutôt voir comment réagissent nos deux modèles si nous intégrons à chaque fois l'ensemble des annotations issues de DICTA, IN_CITATION et NON_CITATION pour tous les seuils en plus des LFs basées sur l'analyse du corpus. Nous avons testé différentes configurations :

— Oxford et HR avec des seuils à 25 (Table 5.10)
— Oxford avec des seuils à 18 et 25 et HR avec un seuil à 25 (Table 5.11)
— HR avec des seuils à 18 et 25 et Oxford avec un seuil à 25 (Table 5.12)

La comparaison des scores ci-dessous avec les scores précédents montre des résultats meilleurs, puisque qu'on gagne dans certains cas quasiment 4 voir 5 points d'accuracy par rapport aux configurations précédentes. Au global, c'est en combinant le MajorityLabelVoter avec HR (seuils a 18 et 25) et Oxford (seuil a 25) qu'on obtient le meilleur score d'accuracy (92%). Ce résultat est intéressant car jusqu'à maintenant le LabelModel surpassait toujours le MajorityLabelVoter.

La comparaison des différentes configurations entre elles montre que seule la configuration avec Oxford et HR avec des seuils à 25 permet au LabelModel d'ob-

TABLE 5.10 – Oxford et HR avec des seuils à 25

MajorityLabelVoter					
Label	**Précision**	**Rappel**	**F1-Score**	**Support**	**Accuracy**
0	0.95	0.94	0.94	57606	
2	0.92	0.66	0.77	17466	0.8764
LabelModel					
Label	**Précision**	**Rappel**	**F1-Score**	**Support**	**Accuracy**
0	0.96	0.92	0.94	57606	0.9081
2	0.77	0.87	0.81	17466	

TABLE 5.11 – Oxford avec des seuils à 18 et 25 et HR avec un seuil à 25

MajorityLabelVoter					
Label	**Précision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.94	0.96	0.95	57606	
2	0.84	0.75	0.79	17466	0.9066
LabelModel					
Label	**Précision**	**Rappel**	**F1-Score**	**Support**	**Accuracy**
0	0.97	0.87	0.92	57606	
2	0.68	0.91	0.78	17466	0.8805

TABLE 5.12 – HR avec des seuils à 18 et 25 et Oxford avec un seuil à 25

MajorityLabelVoter					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.94	0.96	0.95	57606	0.9164
2	0.87	0.76	0.81	17466	
LabelModel					
0	0.97	0.87	0.92	57606	0.8792
2	0.68	0.92	0.78	17466	

tenir un meilleur score d'accuracy, le rappel de la classe 2 chutant à 66% pour le MajorityLabelVoter. L'ajout des annotations avec un seuil à 18 permet au MajorityLabelVoter de surpasser le LabelModel alors que dans les expériences précédentes le LabelModel surpasse le MajorityLabelVoter en terme d'accuracy. On peut penser que cela est dû au fonctionnement du seuil de DICTA. Quand on baisse le seuil, de nouvelles annotations vont être détectées mais celles détectées par les seuils plus haut le reste par les seuils plus bas. Ces seuils vont donc augmenter le nombre de votes pour les citations déjà détectées par les seuils plus hauts. Il faudrait réaliser d'autres expériences pour voir comment réagissent les modèles avec plus de seuils.

Les figures 5.11, 5.12 et 5.13 confirment aussi l'avantage de la configuration qui intègre les annotations complètes de HR avec des seuils à 18 et 25 et Oxford avec un seuil à 25 (figure 5.13)

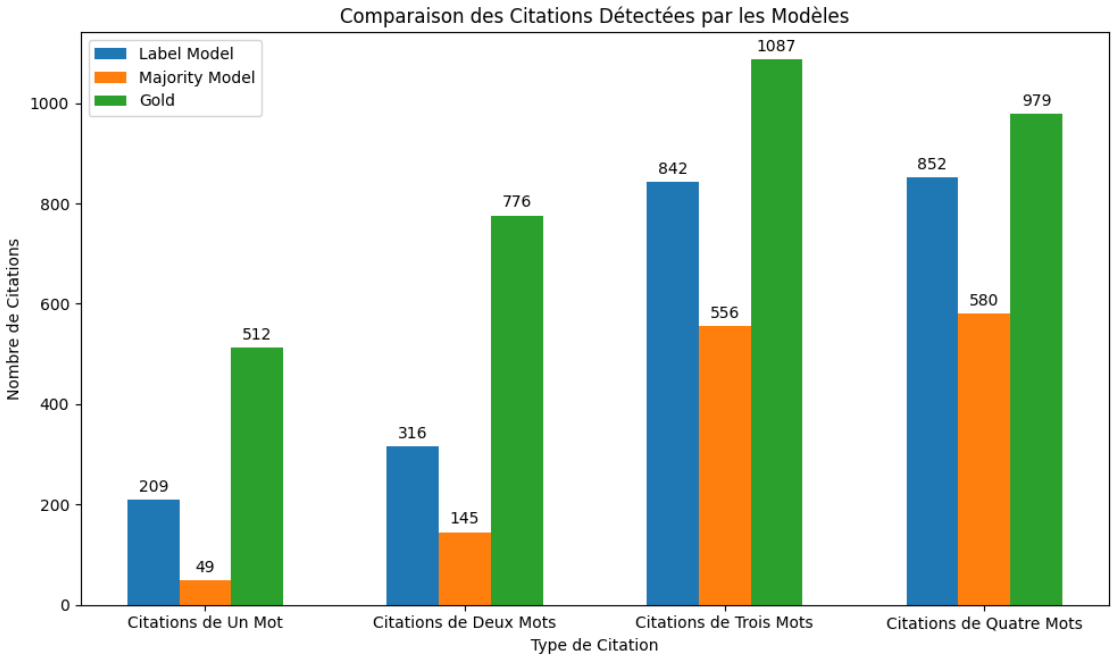

FIGURE 5.11 – Longueurs détectées avec oxf25 et HR25 complets

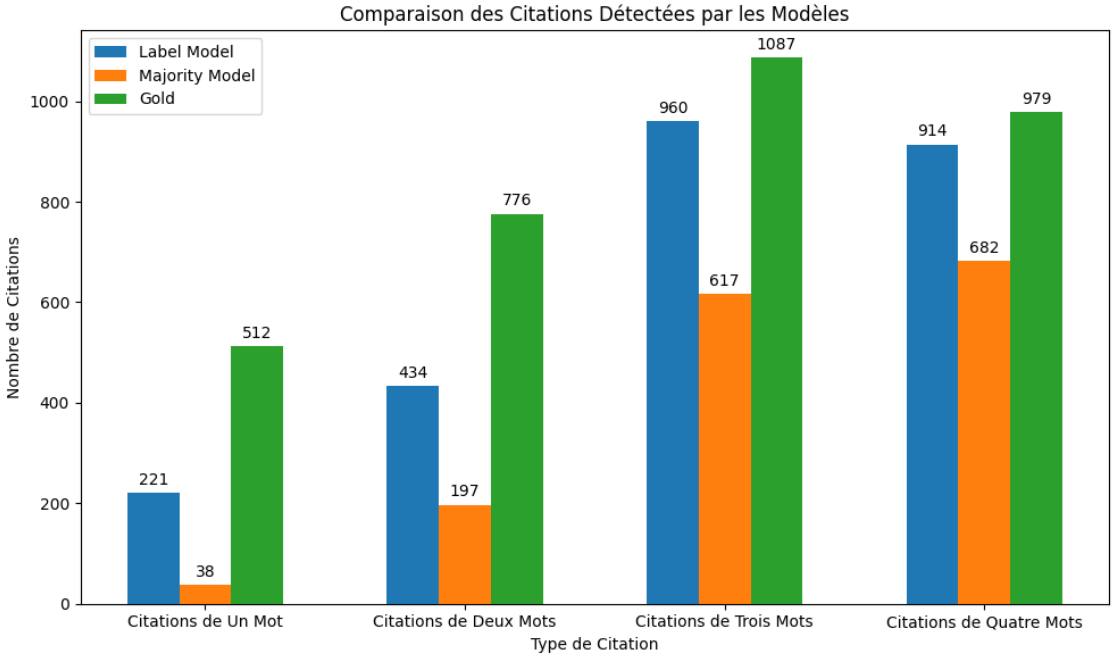

FIGURE 5.12 – Longueurs détectées avec oxf18, oxf25, HR25 complets

5.5.5 On enlève les LFs issues de l'analyse du corpus

Pour tester la contribution des LFs issues de l'analyse du corpus dans le résultat final, nous avons pris la meilleure des configurations précédentes mais en ne gardant que les annotations de DICTA. On voit sur la table 5.13 que la différence n'est pas flagrante. Le score d'accuracy est identique pour le LabelModel, tandis que la perte

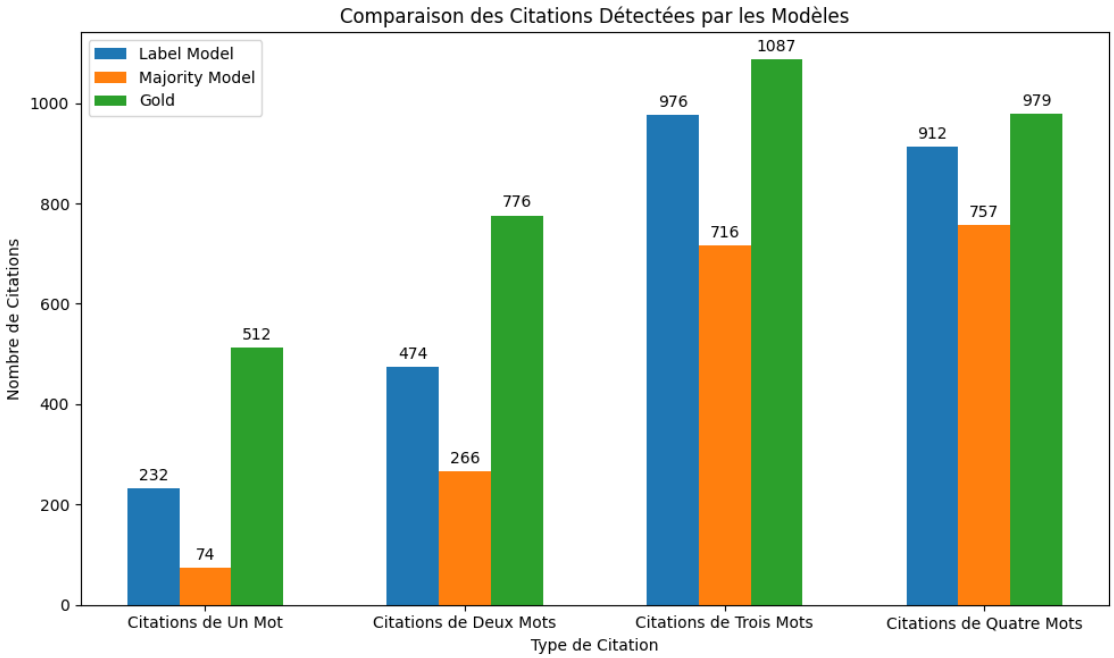

FIGURE 5.13 – Longueurs détectées avec HR18, HR25, oxf25 complets

n'est que de 0.001 sur le MajorityLabelVoter. Le poids des annotations de DICTA dans nos modèles semble écraser les LFs basées sur l'analyse du corpus. Néanmoins, La figure 5.14 montre une chute dans la détection des citations courtes. Malgré une faible différence dans les résultats globaux, les LFs liées à l'analyse du corpus sont donc essentielles pour la détection des citations courtes. Cela montre aussi qu'il pourrait être intéressant de trouver un mode d'évaluation de nos résultats qui permettrait de quantifier plus précisément les variations du taux de détection des citations courtes.

TABLE 5.13 – Dicta25, HR18-25 sans autres LFs

MajorityLabelVoter					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.93	0.97	0.95	57606	0.9154
2	0.87	0.75	0.80	17466	
LabelModel					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.97	0.87	0.92	57606	0.8792
2	0.68	0.92	0.78	17466	

5.5.6 Ajout d'annotations d'Oxford issues de Dicta avec des seuils très bas

Pour cette partie, nous avons choisis de tester l'ajout de deux sets d'annotations obtenus à partir des sorties de DICTA_Oxf paramétrées sur des seuils très bas (10 et 15). Voici la liste des LFs utilisées :

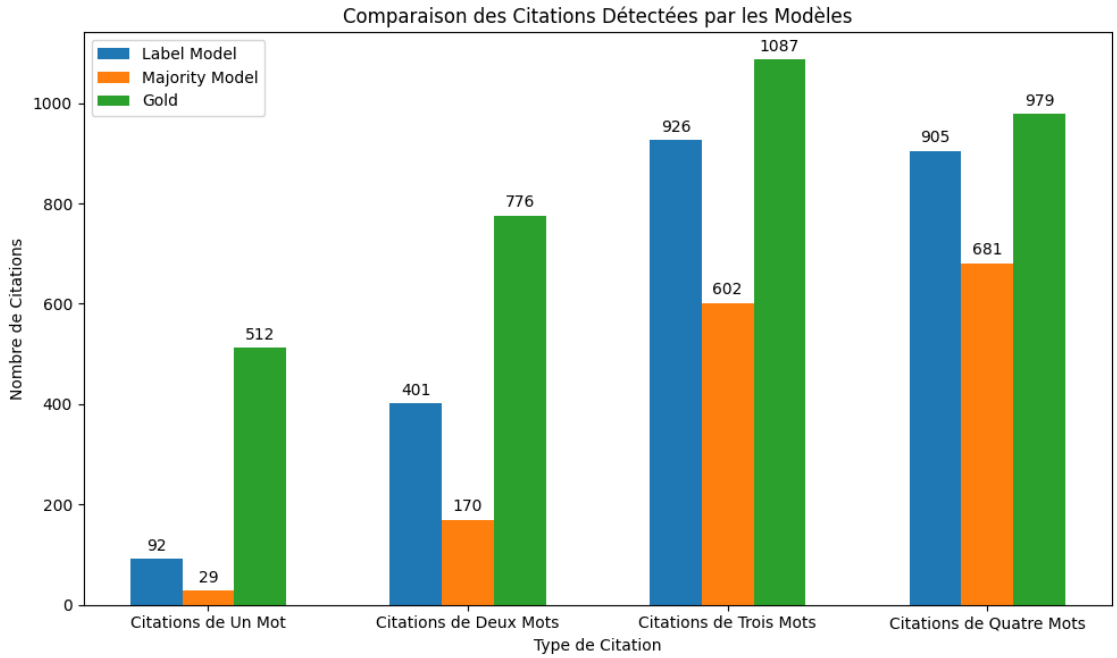

FIGURE 5.14 – Longueurs détectées avec HR18, HR25, oxf25 complets sans autres LFs

```
[LF_morpho,
LF_preceding_word, LF_preceding_bigram, LF_preceding_trigram, LF_next_word,
LF_one_word_cpl, LF_two_words_cpl_first, LF_two_words_cpl_last,
LF_three_words_cpl_first, LF_three_words_cpl_middle,
LF_three_words_cpl_last,
LF_dicta25_IN_CIT, LF_dicta18_NON_CIT,
LF_dictaHR25_IN_CIT,
LF_dicta18_INCIT, LF_dicta15_ALL, LF_dicta10_ALL]
```

TABLE 5.14 – Ajout de Dicta 15 et 10

MajorityLabelVoter avec Dicta 10 et 15					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.98	0.45	0.62	57606	
2	0.59	0.90	0.71	17466	0.5546
LabelModel avec Dicta 10 et 15					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.97	0.87	0.92	57606	
2	0.72	0.80	0.76	17466	0.8544

Les expériences réalisées avec les seuils très bas montrent clairement la supériorité du LabelModel dans des situations très bruitées. Sur la classe 0 l'amélioration est très nette sur le rappel. Sur la classe 2 (IN_CITATION) c'est la précision qui bénéficie le plus du LabelModel. Au final, on voit que le LabelModel réalise un meilleur équilibre entre précision et rappel pour les deux classes, ce qui

permet d'obtenir un score global plus élevé.

Les mauvais résultats du MajorityLabelVoter peuvent être expliqués par la structure des annotations. Lorsqu'il est réglé sur un seuil très bas DICTA va annoter beaucoup de tokens IN_CITATION avec un fort taux de faux positifs. Cela explique pourquoi, concernant la classe 2, le rappel va être très élevé (beaucoup de cas détectés) mais avec une précision faible (beaucoup de faux positifs). Inversement, beaucoup de non-citations vont être annotées IN_CITATION, ce qui explique le rappel faible pour cette classe.

Cette expérience montre que le LabelModel va mieux tirer partie des conflits entre les annotations issues de DICTA avec un seuil élevée et celles avec un seuil faible. Tandis que les résultats du MajorityLabelVoter vont être dégradés par l'ajout de bruit dans les données.

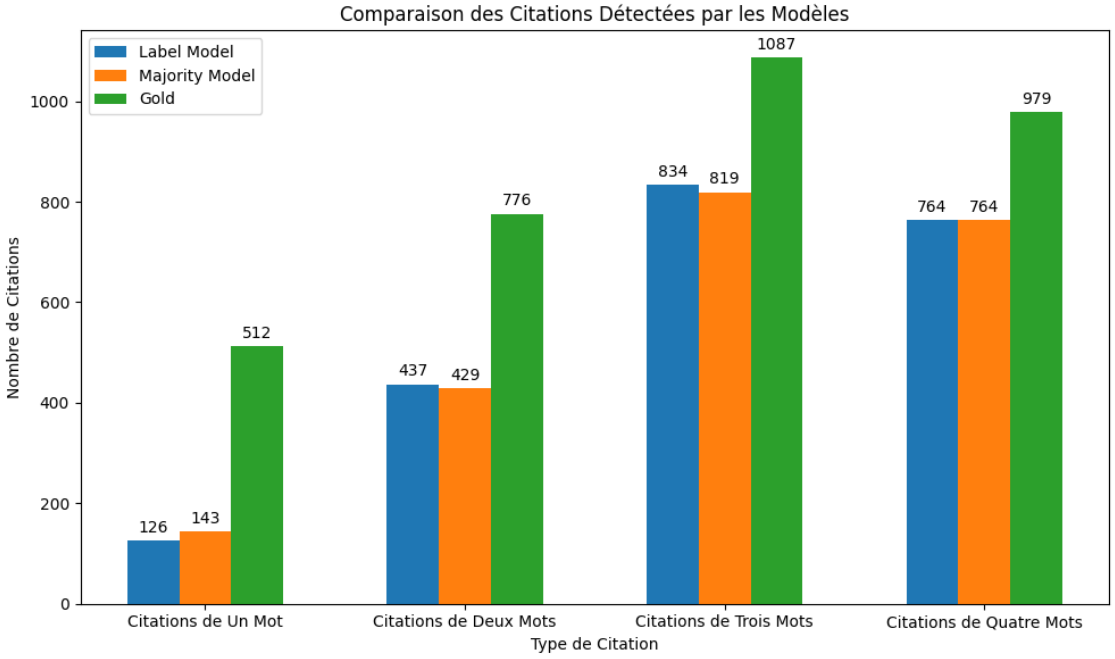

FIGURE 5.15 – Longueurs détectées avec des seuils très bas ajoutés aux autres LFs

La figure 5.15 donne le nombre de citations courtes détectées. Les résultats sont relativement corrects par rapport aux expériences précédentes. On voit que la quantité de citations courtes détectées par le MajorityLabelVoter est beaucoup plus importante que précédemment. Mais ces résultats doivent être mis au regard du fort taux de détection de faux positifs. Beaucoup de citations courtes sont détectées, mais avec une précision faible, ce qui ne rend pas cette configuration très intéressante pour résoudre notre problème. Il est donc impératif de comparer les résultats obtenus par longueur avec la précision globale du modèle.

5.5.7 Résultats avec tous les annotateurs

Regardons maintenant comment réagissent nos deux modèles lorsqu'on ajoute tous les sets d'annotations issues des analyses de DICTA sur Oxford et Horovitz-Rabin à tous les seuils disponibles. Nous avons notamment ajouté des sets d'annotations de DICTA_Oxf avec des seuils à 35 et 40, inédits jusqu'ici. Voici les LFs utilisées :

```
[LF_morpho,
LF_preceding_word, LF_preceding_bigram, LF_preceding_trigram, LF_next_word,
LF_one_word_cpl, LF_two_words_cpl_first, LF_two_words_cpl_last,
LF_three_words_cpl_first, LF_three_words_cpl_middle,
LF_three_words_cpl_last,
LF_dicta25_ALL, LF_dicta18_ALL,
LF_dictaHR25_ALL,  LF_dictaHR18_ALL,
LF_dicta15_ALL, LF_dicta10_ALL,
LF_dicta35_ALL, LF_dicta40_ALL]
```

TABLE 5.15 – Tous les annotateurs de DICTA

MajorityLabelVoter					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.94	0.93	0.93	57606	
2	0.88	0.74	0.80	17466	0.8848
LabelModel					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.98	0.76	0.86	57606	
2	0.59	0.84	0.69	17466	0.7827

Ici c'est le MajorityLabelVoter qui obtient les meilleurs résultats. Le poids d'annotateurs avec des seuils plus élevés explique peut-être ce phénomène. La figure 5.16 montre que le cumul de tous les annotateurs, même en ajoutant des seuils plus élevés, n'améliore pas forcément les résultats. Cela tend à prouver que le processus d'affinement des performances obtenues avec Snorkel relève d'un cheminement itératif d'essais et d'erreurs.

Cela montre aussi que le LabelModel n'est pas toujours la meilleure solution. Ce résultat apporte un élément de réponse à notre question initiale qui était de savoir si la combinaison réalisée par la supervision faible permet d'obtenir de meilleurs résultats que la somme des signaux entre eux. Si l'on considère le MajorityLabelVoter comme une manière de sommer les signaux entre eux, peut-être que la supervision faible telle qu'envisagée avec le LabelModel n'est pas toujours une solution intéressante. Néanmoins, l'utilisation de *Snorkel* montre l'apport d'un tel outil pour combiner des règles entre elles, quelle que soit la qualité des signaux.

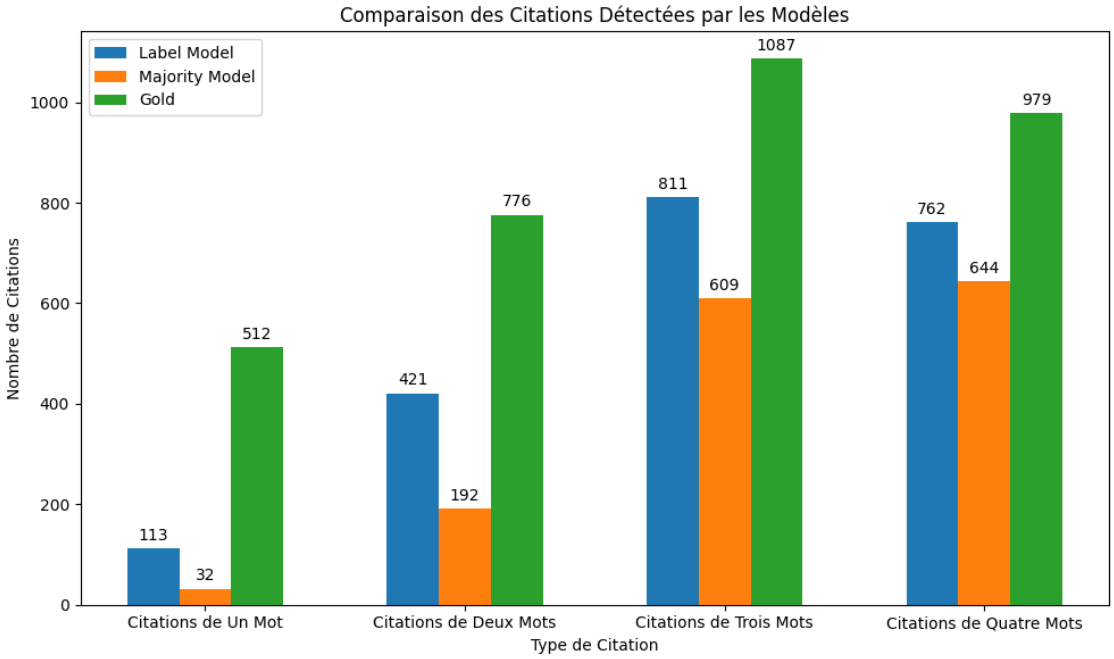

FIGURE 5.16 – Longueurs détectées avec toutes les LFs

Chapitre 6

Résultats

6.1 Résultats obtenus avec Snorkel

Nous allons maintenant voir quelles réponses les résultats de nos expériences apportent aux questions soulevées par l'utilisation de *Snorkel*. Tout d'abord, concernant l'équilibre entre données bruitées et non-bruitées, on voit qu'avec des données très bruitées c'est le LabelModel qui s'en sort le mieux.

Globalement, le LabelModel surpasse souvent le MajorityLabelVoter sur le score d'accuracy, même si on observe parfois un avantage du MajorityLabelVoter sur l'une ou l'autre des classes. Néanmoins, l'expérience qui obtient le score d'accuracy le plus élevé (92%) est réalisée avec le MajorityLabelVoter. Cela montre que le modèle le plus sophistiqué n'est pas toujours le plus performant. Un système de classes trop complexe peut aussi conduire à plus de conflits et une moindre performance globale des modèles.

Il est difficile de répondre de manière définitive à la question de l'équilibre entre données bruitées et non bruitées. L'usage montre que la construction d'un modèle performant avec *Snorkel* passe forcément par un processus d'ajustements continus. On note tout de même une meilleure robustesse du LabelModel avec un mix de données bruitées et non-bruitées.

Les résultats les meilleurs ont été obtenus avec une configuration qui combine les LFs basées sur l'analyse du corpus avec les annotations issues de DICTA sur Oxford (seuil à 25) et DICTA sur Horovitz-Rabin (seuils à 18 et 25). Les tableaux 6.2 et 6.1 nous rappellent les scores obtenus par DICTA_HR25 seul et DICTA_Oxf25 seuls. Le tableau 6.3 expose le score de la configuration la meilleure, qui combine DICTA_HR25, DICTA_HR18 et DICTA_Oxf25 avec les règles basées sur l'analyse du corpus. La comparaison de ces tableaux montre un gain sur le score d'accuracy global, mais aussi sur les autres métriques. La stratégie de récupérer les annotations de DICTA sur Horovitz-Rabin en les alignant avec Oxford s'est avérée efficace. La combinaison avec différents sets d'annotation de DICTA aussi.

On note tout de même que les scores des autres expériences ont rarement dépassé ceux de DICTA25 sur Oxford seul. En effet, l'ajout de nouveaux sets d'annotations augmente le risque de conflits et donc d'erreurs des modèles. L'imprévisibilité des résultats obtenus avec le LabelModel oblige donc à un certain tâtonnement. Il est

donc plutôt recommandé d'avoir de nombreuses sources à combiner pour pouvoir tester différentes combinaisons. Les premières expériences montrent que la simple combinaison de nos LFs basées sur l'analyse du corpus avec les analyses de DICTA sur Oxford est insuffisante.

TABLE 6.1 – Rapports de classification pour Oxf DICTA seuil à 25

Dicta seuil 25					
Label	**Précision**	**Recall**	**Score F1**	**Support**	**Accuracy**
0	0.91	0.96	0.93	57606	
2	0.84	0.68	0.75	17466	0.8952

TABLE 6.2 – Rapports de classification pour HR DICTA seuil à 25

Dicta seuil 25					
Label	**Précision**	**Recall**	**Score F1**	**Support**	**Accuracy**
0	0.92	0.97	0.94	57606	
2	0.87	0.73	0.80	17466	0.9133

TABLE 6.3 – HR avec des seuils à 18 et 25 et Oxford avec un seuil à 25

MajorityLabelVoter					
Label	**Precision**	**Recall**	**F1-Score**	**Support**	**Accuracy**
0	0.94	0.96	0.95	57606	0.9164
2	0.87	0.76	0.81	17466	
LabelModel					
0	0.97	0.87	0.92	57606	0.8792
2	0.68	0.92	0.78	17466	

6.2 Pistes pour améliorer le modèle

L'utilisation de *Snorkel* oblige donc à une certaine ingéniosité pour trouver des moyens d'améliorer les performances du modèle, ce qui nous a poussé à imaginer différentes pistes pour améliorer nos résultats.

Tout d'abord, un premier constat pourrait être que nos premières LFs basées sur l'analyse du corpus étaient peut être un peu trop précises. Une idée serait alors de bruiter ces LFs volontairement en ajoutant dans nos listes des ngrams précédents ou suivants qui n'apparaissent pas toujours dans le contexte de citations. D'un autre côté, il serait intéressant aussi de pouvoir donner plus de poids aux LFs dont on connait à l'avance la fiabilité. Cependant, le model de *Snorkel* part du principe que la fiabilité des LFs n'est pas connue à l'avance et va alors déterminer des probabilités aux étiquettes fournies par les LFs en effectuant des recoupements.

On peut aussi imaginer écrire des règles plus complexes qui exploiteraient les interactions entre nos LFs de base et les annotations de DICTA. Par exemple, nous avons constaté que DICTA manquait parfois le début ou la fin des citations. La

combinaison avec les autres LFs pourrait permettre de reconstruire l'ensemble de certaines citations, d'autant plus que, comme nous l'avons observé, les interactions entre LFs peuvent conduire à des conflits dommageables aux performances du modèle. Cela pourrait également aider à résoudre les problèmes liés aux différences de couverture entre LFs.

La proposition précédente pourrait aussi être réalisée en pré-traitement avant d'alimenter le modèle ou en post-traitement pour ajuster et enrichir les prédictions. L'idée est que comme *Snorkel* peut améliorer des annotations externes, mais qu'il peut aussi les dégrader, on peut imaginer améliorer ses prédictions à l'aide de pré/posts-traitement qui exploiteraient des caractéristiques dont on est sûrs de la précision. Nous avons vu que la multiplicité des tags avait posé problème avec les annotations de DICTA. Avec des pré-traitements nous pourrions aussi créer un set d'annotations hybride qui combinerait à l'avance les annotations de DICTA et la détection des premiers et derniers mots des citations. De cette façon, nous pourrions réhabiliter notre schéma d'annotation à six classes avec DICTA.

Nous n'avons pas non plus exploité les répétitions de ngrams de citations dans ce travail. Des expériences préliminaires ont montré que cette technique pouvait fonctionner mais avec une précision de 50% environ et pour une couverture très faible. En plus, nous pourrions aussi définir des LFs basées sur des lexiques de mots apparaissant plutôt dans les citations ou plutôt dans les non-citations. Nous avons envisagé ces pistes au départ, en les abandonnant ensuite car la précision était trop faible. Il pourrait néanmoins être intéressant de tester ces caractéristiques avec le LabelModel car il est robuste face au manque de précision.

Comme nous l'avons vu dans le chapitre 4, d'autres méthodes heuristiques pourraient être envisagées, notamment en exploitant la structure de la *Mekhilta*. À la manière de DICTA, il s'agirai de chercher des correspondances de ngrams, mais en restreignant la cible au Livre ou au chapitre de la Bible étudié dans le passage en cours, ce qui permettrait de réduire la fréquence et donc d'augmenter la précision du système.

Toutes ces propositions concernent l'exploitation des caractéristiques des données. Une autre stratégie serait d'optimiser *Snorkel* en utilisant d'autres fonctionnalités telle que l'augmentation de données, technique qui pourrait nous permettre de rééquilibrer nos classes. Des paramétrages plus fins sont également possibles sur le LabelModel.

Un autre aspect concerne l'optimisation des expériences en vue des résultats que l'on cherche à obtenir. Les citations les plus longues pèsent beaucoup plus que les citations courtes car elles représentent plus de tokens, ce qui leur donne plus de poids. D'autres types de découpages pourraient donner plus de poids aux citations courtes.

Conclusion générale

Dans ce mémoire, nous avons abordé la question de la constitution d'un corpus annoté au travers de l'utilisation d'un outil de supervision faible. La problématique était de savoir si les performances obtenues permettrait de dépasser les résultats obtenus avec DICTA et si cette technique s'avère plus efficace que l'utilisation simple de règles sans supervision faible. Autrement dit, est-il utile d'utiliser ou outil comme *Snorkel* lorsqu'on a des sources plutôt fiable, ou seulement lorsqu'on a des signaux bruités.

Notre travail nous a conduit à utiliser des méthodes statistiques pour analyser le corpus, mais aussi des techniques pour associer différentes sources, comme nous l'avons fait en alignant le manuscrit d'Oxford avec l'édition d'Horovitz-Rabin. *Snorkel* s'est révélé très efficace dans la combinaison de ces différentes sources de qualités variables.

Nos expériences ont aussi démontré que *Snorkel* pouvait être un outil intéressant à utiliser aussi bien dans les cas d'informations précises que d'informations bruités. Les deux modèles proposés par le système pouvant s'adapter à ces différentes configurations, le LabelModel révèlant toutes ses capacités dans des situations où les informations sont très bruitées.

Globalement, notre travail prouve aussi l'utilité d'un outil comme *Snorkel* dans les situations contraintes par le manque de données et de moyens, fréquentes dans les projets de recherche en TAL et en Humanités Numériques, surtout lorsqu'on s'intéresse à des langues peu étudiées comme l'hébreu rabbinique.

Cette étude nous a aussi permis d'explorer en profondeur notre corpus de littérature rabbinique. Ainsi, l'exploration de pistes fructueuses et infructueuses nous a permis de mieux comprendre ce type de texte, ce qui nous sera très utile pour la suite. L'analyse de la *Mekhilta de Rabbi Yishmael* n'est qu'une première étape dans notre projet dont l'ambition est de développer un outil capable de détecter finement les citations quelles que soient leurs longueurs et pour tout type de texte rabbinique. Il va donc s'agir de trouver une méthode apte à généraliser par delà la *Mekhilta*. L'évaluation de méthodes heuristiques était la première étape. La suite de notre travail consistera à utiliser le corpus constitué pour entraîner un modèle d'apprentissage automatique. Dans cette nouvelle quête, le modèle de langue pré-entraîné sur l'hébreu rabbinique BEREL [Shmidman et al., 2022] constituera un élément central de nos expériences.

Bibliographie

[Augenstein et al., 2015] Augenstein, I., Vlachos, A., and Maynard, D. (2015). Extracting relations between non-standard entities using distant supervision and imitation learning. In Màrquez, L., Callison-Burch, C., and Su, J., editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 747–757, Lisbon, Portugal. Association for Computational Linguistics. – Cité page 29.

[Benamar, 2020] Benamar, A. (2020). Segmentation de texte non-supervisée pour la détection de thématiques à l'aide de plongements lexicaux (unsupervised text segmentation for topic detection using embeddings). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 3 : Rencontre des Étudiants Chercheurs en Informatique pour le TAL*, pages 1–14, Nancy, France. ATALA et AFCP. – Cité page 26.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, page 92–100, New York, NY, USA. Association for Computing Machinery. – Cité page 27.

[Caragea et al., 2015] Caragea, C., Bulgarov, F., and Mihalcea, R. (2015). Co-training for topic classification of scholarly data. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2357–2366, Lisbon, Portugal. Association for Computational Linguistics. – Cité pages 27 et 28.

[Caruana, 1993] Caruana, R. (1993). Multitask learning : A knowledge-based source of inductive bias. In *International Conference on Machine Learning*. – Cité page 29.

[Einat-Nov, 2013] Einat-Nov, I. (2013). Esthetic qualities, conventions, and aspect-switching : Medieval hebrew poetry in the perspective of modern theories of reading. *Style*, 47(1) :55–68. – Cité page 22.

[Franzini, 2016] Franzini, G., F. E. B. M. (2016). "historical text reuse : What is it ?.". http ://www.etrap.eu/historical-text-re-use/. – Cité pages 18, 18, 19 et 19.

[Heck et al., 2022] Heck, M., Lubis, N., van Niekerk, C., Feng, S., Geishauser, C., Lin, H.-C., and Gašić, M. (2022). Robust dialogue state tracking with weak supervision and sparse data. *Transactions of the Association for Computational Linguistics*, 10 :1175–1192. – Cité page 30.

[Karamanolakis et al., 2021] Karamanolakis, G., Mukherjee, S., Zheng, G., and Awadallah, A. H. (2021). Self-training with weak supervision. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, pages 845–863, Online. Association for Computational Linguistics. – Cité page 30.

[Lee, 2007] Lee, J. (2007). A computational model of text reuse in ancient literary texts. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 472–479, Prague, Czech Republic. Association for Computational Linguistics. – Cité page 17.

[Lee and Chieu, 2021] Lee, J. Y. D. and Chieu, H. L. (2021). Co-training for commit classification. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 389–395, Online. Association for Computational Linguistics. – Cité page 28.

[Lison et al., 2021] Lison, P., Barnes, J., and Hubin, A. (2021). skweak : Weak supervision made easy for NLP. In Ji, H., Park, J. C., and Xia, R., editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing : System Demonstrations*, pages 337–346, Online. Association for Computational Linguistics. – Cité page 30.

[Maamatou, 2017] Maamatou, H. (2017). *Apprentissage semi-supervisé pour la détection multi-objets dans des séquences vidéos : Application à l'analyse de flux urbains*. PhD thesis. – Cité page 27.

[MacLaughlin et al., 2021] MacLaughlin, A., Xu, S., and Smith, D. A. (2021). Recovering lexically and semantically reused texts. In *Proceedings of *SEM 2021 : The Tenth Joint Conference on Lexical and Computational Semantics*, pages 52–66, Online. Association for Computational Linguistics. – Cité pages 17 et 18.

[Moritz and Büchler, 2017] Moritz, M. and Büchler, M. (2017). Ambiguity in semantically related word substitutions : an investigation in historical Bible translations. In *Proceedings of the NoDaLiDa 2017 Workshop on Processing Historical Language*, pages 18–23, Gothenburg. Linköping University Electronic Press. – Cité page 17.

[Moritz et al., 2016] Moritz, M., Wiederhold, A., Pavlek, B., Bizzoni, Y., and Büchler, M. (2016). Non-literal text reuse in historical texts : An approach to identify reuse transformations and its application to Bible reuse. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1859, Austin, Texas. Association for Computational Linguistics. – Cité page 17.

[Nelson, 1999] Nelson, W. D. (1999). The reconstruction of the "mekhilta of rabbi shimon b. yoḥai" : A reexamination. *Hebrew Union College Annual*, 70/71 :261–302. – Cité page 36.

[Niculae et al., 2015] Niculae, V., Suen, C., Zhang, J., Danescu-Niculescu-Mizil, C., and Leskovec, J. (2015). QUOTUS : the structure of political media coverage as revealed by quoting patterns. *CoRR*, abs/1504.01383. – Cité page 18.

[Qazvinian and Radev, 2010] Qazvinian, V. and Radev, D. R. (2010). Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 555–564, Uppsala, Sweden. Association for Computational Linguistics. – Cité page 18.

[Ratner et al., 2017] Ratner, A., Bach, S. H., Ehrenberg, H. R., Fries, J. A., Wu, S., and Ré, C. (2017). Snorkel : Rapid training data creation with weak supervision. *CoRR*, abs/1711.10160. – Cité pages 30 et 53.

[Shmidman, 2022] Shmidman, A. (2022). *Automatic Identification of Biblical Citations and Allusions in Hebrew Texts*, pages 335–348. De Gruyter Oldenbourg, Berlin, Boston. – Cité pages 21, 43 et 46.

[Shmidman et al., 2022] Shmidman, A., Guedalia, J., Shmidman, S., Shmidman, C. S., Handel, E., and Koppel, M. (2022). Introducing berel : Bert embeddings for rabbinic-encoded language. – Cité pages 29, 29 et 83.

[Stemberger and Bockmuehl, 1996] Stemberger, G. and Bockmuehl, M. N. A. (1996). *Introduction to the Talmud and Midrash*. TT Clark, Edinburgh, 2nd ed edition. – Cité pages 36, 36 et 36.

[Wan, 2009] Wan, X. (2009). Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243, Suntec, Singapore. Association for Computational Linguistics. – Cité page 28.

[Wilkerson et al., 2015] Wilkerson, J., Smith, D., and Stramp, N. (2015). Tracing the flow of policy ideas in legislatures : A text reuse approach. *American Journal of Political Science*, 59(4) :943–956. – Cité page 18.

[Yuan et al., 2020] Yuan, M., Lin, H.-T., and Boyd-Graber, J. (2020). Cold-start active learning through self-supervised language modeling. In Webber, B., Cohn, T., He, Y., and Liu, Y., editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7935–7948, Online. Association for Computational Linguistics. – Cité page 28.

[Zhang et al., 2022] Zhang, Z., Strubell, E., and Hovy, E. (2022). A survey of active learning for natural language processing. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6166–6190, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics. – Cité page 28.