Détection de doublons parmi des informations non structurées provenant de sources de données différentes

Mémoire

David Beauchemin

Maîtrise en informatique - avec mémoire

Maître ès sciences (M. Sc.)

Québec, Canada

Détection de doublons parmi des informations non structurées provenant de sources de données différentes

Mémoire

David Beauchemin

Sous la direction de:

Luc Lamontagne, directeur de recherche

Résumé

Ce mémoire rend compte de l'exploration de deux approches de détection des doublons entre les descriptions d'entreprises d'une base de données interne et celles d'une source externe non structurée en assurance commerciale. Puisqu'il est coûteux et fastidieux pour un assureur de recueillir les informations nécessaires au calcul d'une prime d'assurance, notre motivation est de les aider à minimiser la quantité de ressources nécessaires à leur acquisition en leur permettant d'utiliser des sources de données externes.

Dans ce mémoire, nous avons d'abord observé que l'utilisation d'algorithmes de similarité permet de détecter la majorité des doublons entre les sources de données à partir du nom. Nos expérimentations indiquent que lorsqu'on utilise le nom comme source de comparaison entre les entités, une très grande majorité de ces doublons peut être identifiée. Des expérimentations similaires, mais avec l'adresse, nous ont permis d'observer qu'il était aussi possible d'identifier les doublons d'entreprises par cet attribut, mais dans une moins grande proportion. Par la suite, nous avons entraîné des modèles d'apprentissage automatique afin de coupler les entreprises en double par le nom et l'adresse conjointement. C'est avec ces modèles que nous avons observé les meilleurs résultats. Dans une tentative finale d'améliorer davantage nos résultats, nous avons assoupli notre hypothèse initiale, qui impliquait d'utiliser l'entité la plus probable d'être le doublon d'une entreprise, pour utiliser les N entités les plus probables, ce qui a permis de maximiser le rappel à 91,07 %.

Abstract

This thesis reports the exploration of two approaches to detecting duplicates between the companies descriptions in an internal database and those in an unstructured external source in commercial insurance. Since it is costly and tedious for an insurer to collect the information required to calculate an insurance premium, our motivation is to help them minimize the amount of resources necessary by extracting that information directly from external databases.

In this thesis, we first observed that the use of similarity algorithms allows us to detect most of the duplicates between databases using the name. Our experiments indicate that when the name is used as a source of comparison between the entities, a vast majority of these duplicates can be identified. Similar experiments, but using the address this time, allowed us to observe that it was also possible to identify duplicate companies by this feature, but to a lesser extent. Subsequently, we trained machine learning models to match duplicate companies using the name and the address at the same time. It is with these models that we observed the best results. In a final attempt to further improve our results, we used the N most likely entities to be a duplicate of a company, instead of only the first one, thus maximizing the recall to 91.07%.

Table des matières

Liste des tableaux

Liste des figures

Liste des algorithmes

À Cheryl.

If you will it, it is no dream...

Theodor Herzl

Remerciements

Je remercie mon directeur de recherche, Luc Lamontagne, de m'avoir accepté dans son équipe. Très patient, il a su me guider dans mes études de cycles supérieures malgré mes nombreux projets en parallèle. Par nos discussions, il m'a fait réévaluer de nombreuses décisions, m'a appris à remettre mes choix en question et m'a permis de m'épanouir durant mon parcours.

Je remercie Cheryl Collerette qui a su être patiente et m'épauler dans mes décisions malgré l'adversité. Je la remercie aussi d'avoir été présente malgré mon flagrant manque de disponibilité physique et intellectuelle. Je remercie mes parents de m'avoir donné les bons outils pour me permettre de m'accomplir dans la vie et par le fait même dans mes études de cycles supérieurs. Je remercie aussi mon oncle François Beauchemin et son conjoint Benoît Simard de m'avoir accueilli à Québec et d'avoir été de bons (excellents) premiers colocs. Merci à Benoît pour l'aide à la correction et les commentaires de rédaction.

Je remercie tous les membres du GRAAL que j'ai eu l'honneur de côtoyer durant mon parcours. Merci à Jean-Thomas Baillargeon, Nicolas Garneau, Mathieu Godbout, Jean-Samuel Leboeuf, Gaël Letarte, Frédérik Paradis, Dominique Pothier et Marouane Yassine. Vous avez tous contribué, d'une façon ou d'une autre, à ma formation. De plus, j'aimerais souligner la contribution de Marouane Yassine dans la réalisation de DeepTagger qui m'a été d'une grande utilité.

J'aimerais remercier Intact Corporation financière pour son support financier ainsi que d'avoir mis à ma disposition un jeu de données m'ayant permis de travailler sur une problématique.

Finalement, j'aimerais remercie Daphné Allard-Gervais, Nassim Benchabane, Éliane Bélisle et Simon Martin pour vos conseils et révisions.

Introduction

Ce mémoire porte sur l'utilisation d'informations non structurées en assurance, et plus particulièrement en assurance commerciale. Lorsque qu'une entreprise désire se procurer un service d'assurance commerciale auprès d'un assureur, ce dernier doit faire une analyse en profondeur du risque que représente cette entreprise. Ce processus, appelé souscription du risque, commence par une cueillette intensive de données à propos du client potentiel afin de constituer une proposition d'assurance commerciale (Chair *et al.*, 1997). À partir de cette proposition, l'assureur est en mesure d'apprécier la nature du risque que représente l'entreprise pour ensuite déterminer une prime permettant d'assurer celle-ci (Pollard, 1990).

Il est donc important que la représentation du risque commercial soit la plus précise possible afin de permettre la tarification la plus juste. Pour l'assureur, cette prime doit modéliser fidèlement le risque financier réel que représente l'entreprise, puisqu'une sous-évaluation (c.-à-d. une prime inférieure à la valeur du risque) pourrait avoir des conséquences néfastes sur ses états financiers au moment d'une réclamation (p. ex. mise en péril de sa santé financière). Toutefois, l'inverse serait aussi néfaste pour l'assureur, puisqu'une surévaluation du risque ne lui permettrait pas d'être compétitif sur le marché et pourrait entraîner la perte de clients potentiels (Werner et Modlin, 2010). Il a donc un intérêt à se doter d'un processus efficace de souscription des risques.

Ce type de processus d'évaluation des entreprises est propre à chaque assureur, puisqu'il se base largement sur son expérience dans les différents types de risques présents sur le marché (Desrosiers, 2013). Toutefois, les assureurs utilisent tous des méthodes similaires pour accomplir certaines étapes de l'appréciation du risque. Par exemple, pour remplir une proposition d'assurance, tous doivent obtenir un maximum d'informations auprès de l'entreprise afin d'être en mesure d'en modéliser le risque. Cette étape de collecte de données est cruciale, mais aussi très dispendieuse. En effet, historiquement, les coûts d'exploitation annuelle des assureurs sont d'environ 30 % de leurs revenus (Insurance Bureau of Canada, 2019). Ces importants frais s'expliquent par diverses obligations financières, tels que les coûts d'infrastructures, d'opérations et de personnel. Soulignons notamment qu'une méthode de collecte des données couramment utilisée en assurance commerciale est le questionnaire client. Cette étape est souvent effectuée à travers un réseau de courtiers, ce qui implique des frais de commission. Une visite sur les

lieux d'un souscripteur peut également s'avérer nécessaire si le risque est trop complexe pour un simple questionnaire papier, ce qui ajoute encore des frais au processus d'évaluation du risque. En somme, le processus d'évaluation des entreprises est long et fastidieux pour le client, et dispendieux pour la compagnie d'assurance.

L'utilisation de sources externes de données pour effectuer de manière autonome la cueillette des renseignements sur le risque commercial devient donc attrayante pour une compagnie d'assurance : elle lui permet de réduire ses coûts durant son processus d'appréciation des risques et potentiellement d'améliorer sa représentation du client potentiel. De plus, le recours à ce type de ressource peut aussi contribuer à réduire le nombre de questions dans sa proposition d'assurance, ce qui simplifie du même coup le processus pour le client. Toutefois, l'utilisation d'informations provenant d'une source externe est une tâche difficile à accomplir puisque la représentation des données provenant de l'externe n'est pas forcément la même que celle utilisée par l'assureur. Cette divergence de la modélisation de l'information complexifie le processus de fusion des sources (Christen, 2012b). Pour permettre l'appariement de ces sources, il est nécessaire d'utiliser un algorithme de détection intersources de doublons afin de pouvoir relier l'information sur une même entreprise présente dans les différentes sources de données (Naumann et Herschel, 2010). Cet algorithme permet de mettre en évidence les doublons susceptibles de faire référence à la même entité commerciale. Par exemple, l'entreprise « Les Entreprises Beauchemin », située au « 759, avenue Myrand, Québec, QC, G1V 2V3 », est susceptible d'être la même entreprise que « Entreprises Beauchemin inc. », situé au « 759, avenue Myrand, app. 2, Québec, QC, G1V 2V3 ». Par contre, elle n'est pas susceptible d'être la même entité commerciale que « BD associés », située au « 4247, rue Stéphanie, Laval, QC, H7R 6J1 », puisque leurs noms sont différents et qu'elles sont situées à environ 300 km l'une de l'autre.

La problématique

La problématique que nous abordons dans ce mémoire est la détection des doublons de descriptions d'entités commerciales entre une base de donnée interne et une source externe non structurée. Cette correspondance permet de mettre en évidence les doublons intersources qui font probablement référence à la même entreprise. Toutefois, pour résoudre cette problématique, on fait face à trois difficultés : le type de données, leur qualité et la complexité algorithmique du problème.

Premièrement, les données textuelles non structurées sont difficiles à utiliser, car leur représentation est souvent moins facile à traiter pour un système informatique (Christen, 2012b). Puisque ce type de données est souvent composé d'un dictionnaire de mots très étendu, la représentation d'une simple phrase reste relativement accessible, mais lorsqu'on tente de représenter un paragraphe ou un document en entier, le problème se complexifie rapidement puisqu'il est difficile de comparer un tel nombre de représentations entre elles. En effet, pour

pouvoir effectuer des opérations mathématiques (p. ex. addition, soustraction) sur des données textuelles, il est nécessaire de les transformer. Autrement dit, il faut encoder le texte dans un format permettant ce type d'opération. Notamment, le sac de mots qui comptabilise les mots et leur nombre d'occurrences dans un texte ou encore l'encodage 1 parmi n qui permet d'encoder les mots présents dans une phrase par rapport à un dictionnaire en attribuant la valeur 1. Par exemple, l'encodage de la phrase « Le chat » avec le dictionnaire {Le, La, chien, chat} permet d'obtenir le vecteur $[1, 0, 0, 1]$. Toutefois, ces méthodes d'encodage souffrent d'un problème de grande cardinalité puisqu'une langue est composée d'un vocabulaire très vaste, souvent des milliers de mots. Cette dimensionnalité importante crée des représentations peu denses qui sont plus difficilement représentables pour un algorithme (Bellman, 1966; Zekić-Sušac *et al.*, 2014). Récemment cependant, des algorithmes de modélisation de la langue dans un format plus dense et de moins grande dimensionnalité ont été proposés. Mentionnons notamment les algorithmes de plongement de mots comme Word2Vec (Mikolov *et al.*, 2013a) et GloVe (Pennington *et al.*, 2014a). Ces algorithmes permettent de représenter dans un espace vectoriel de dimension M la relation entre les mots. Typiquement, M est de dimension beaucoup plus faible que la taille du vocabulaire, soit entre 100 et 300. Ainsi, il est possible d'effectuer des opérations mathématiques tout en capturant la relation entre les mots. L'intuition de cette approche est qu'il existe une relation linéaire entre les mots. Par exemple, la relation linéaire entre « roi » et « reine » est le genre. Ainsi, le résultat linéaire de l'opération roi $-$ homme $+$ femme correspond à « reine ». Par contre, ces algorithmes de modélisation de la langue se sont avérés limités dans la résolution de notre problématique puisque ce type de représentation est construit à partir d'un corpus de noms communs et que les données textuelles de notre problème sont principalement constituées de noms propres (Mudgal *et al.*, 2018).

Deuxièmement, puisque les données textuelles sont souvent générées par un utilisateur, elles peuvent contenir des informations de mauvaise qualité. Nous pouvons notamment penser à un agent d'assurance qui inscrirait une information dans le mauvais attribut ou qui ferait une faute dans l'écriture d'un nom en complétant le formulaire de souscription d'un client au téléphone [1]. Que ces erreurs soient de nature typographique, de compréhension sonore ou autres, elles rendent notre problématique encore plus difficile puisque la qualité des données influence grandement la performance des algorithmes (Tayi et Ballou, 1998; Sessions et Valtorta, 2006).

Troisièmement, la complexité algorithmique du problème rend le processus de couplage des données très long. Une approche naïve pour résoudre ce problème est la comparaison de chaque entité de la première source (A) avec l'ensemble de la seconde source (B). Toutefois, si les sources A et B contiennent n données, nous allons avoir $n \times n$ comparaisons à faire, soit une complexité en $\mathcal{O}(n^2)$ (Naumann et Herschel, 2010). Si n est petit, le temps pour effectuer

1. À noter que nous n'avons pas utilisé de données provenant de compagnie d'assurance, nous avons seulement modélisé le problème dans le contexte dans lequel la solution sera utilisée.

ces opérations peut être raisonnable, mais pour des bases de données contenant des millions d'enregistrements, l'approche devient extrêmement longue. Il faut alors se limiter à un nombre réduit de comparaisons présélectionnées pour diminuer le temps d'exécution de l'algorithme (Naumann et Herschel, 2010). Toutefois, si nous limitons trop le nombre de comparaisons, nous pourrions empêcher la détection de doublons valides et réduire le nombre de risques commerciaux correctement couplés.

Autrement dit, la complexité d'établir la correspondance des descriptions d'entités commerciales entre les bases de données vient du fait que les données en cause sont difficiles à représenter, de mauvaise qualité, et que la complexité algorithmique du problème nécessite un effort calculatoire important. Pour y remédier, il sera nécessaire d'utiliser des algorithmes permettant de représenter efficacement les données textuelles tout en fonctionnant bien dans un contexte de données bruitées et ayant une empreinte computationnelle basse.

Structure du document

Ce mémoire de maîtrise est structuré comme suit : le chapitre 1 présente une revue de la littérature pertinente et une définition des principaux concepts abordés. Le chapitre 2 présente les sources de données ainsi que les deux catégories d'algorithmes utilisés pour déterminer la similarité entre une description d'entreprise et celles d'une entité de la source externe à partir de données textuelles. Le chapitre 3 présente les résultats du couplage du nom des entreprises avec les entités de la source externe en utilisant des mesures de similarités. Nous y introduisons également une approche permettant d'améliorer les résultats observés à l'aide d'un seuil minimal de similarité. Le chapitre 4 présente les résultats du couplage de l'adresse. De plus, nous y introduisons une méthode permettant d'utiliser chaque composante de l'adresse plutôt que l'adresse complète. Le chapitre 5 présente les résultats du couplage simultané du nom et de l'adresse en utilisant des algorithmes d'apprentissage automatique ainsi qu'une version assouplie de l'algorithme de couplage. Finalement, ce mémoire se termine par une discussion sur les résultats obtenus, les limites de ceux-ci et une réflexion sur les améliorations possibles.

Chapitre 1

Revue de littérature

Ce premier chapitre présente les concepts importants de ce mémoire ainsi que des algorithmes de détection de doublons.

1.1 Faire correspondre les entités

Les sources externes de données sont une ressource intéressante pour une compagnie d'assurance puisqu'elles peuvent fournir une représentation d'un risque commercial de manière autonome et potentiellement à moindre coût. Toutefois, pour bénéficier de son utilisation, il est nécessaire de recourir à un algorithme de détection de doublons afin d'extraire des informations sur des entreprises provenant de la source externe de données. Ce couplage des entités nécessite une méthode permettant de déterminer la similarité entre celles-ci à partir de données textuelles (Winkler, 1999). Le calcul de similarité est basé sur différents attributs communs entre les entités (Christen, 2012b). Par exemple, leur nom, leur date de création, leur propriétaire ou toute autre information partagée entre les sources de données tel qu'établi avec le partenaire. Par contre, dans notre cas, seulement deux attributs communs sont disponibles : le nom de l'entreprise et son adresse. Ce nombre limité d'informations communes entre les entités est dû à deux facteurs, un volontaire et un autre contextuel. Volontaire d'abord, parce que l'un des objectifs est de réduire le nombre de questions dans le questionnaire de la proposition d'assurance. Cette limitation a donc été incorporée dans la problématique et nous nous limitons donc à l'utilisation de ces deux attributs. Contextuelle ensuite, parce que comme le démontre l'exemple de questionnaire à l'annexe A, les informations sont soit de nature non publiques (p. ex. état financier), très spécifiques (p. ex. pourcentage des actifs par pays) ou temporellement variables (p. ex. nombre d'employés). Toutes ces caractéristiques rendent difficile le recoupement de ces informations avec des sources externes.

Plusieurs algorithmes de classification permettent de déterminer la similarité entre les attributs textuels d'une entreprise et ceux d'une entité de la source externe. Ces algorithmes sont divisés en deux groupes : par comparaison d'attributs et par algorithme de classification.

1.1.1 Comparaison d'attributs

La première catégorie d'algorithmes compare les attributs partagés par les deux entités afin de déterminer si leur référent est la même entreprise (Christen, 2012c). Dans notre contexte, cela revient à comparer les noms ou les adresses des entreprises afin de déterminer s'ils sont correspondants. Par exemple, si elles partagent le même code postal et la même date de création, alors ils représentent la même entité commerciale. Pour comparer les attributs, nous utilisons notamment des mesures de similarité à base de chaînes de caractères (Cohen *et al.*, 2003b). Ces algorithmes de comparaison textuelle permettent de mesurer la similarité entre deux chaînes de caractères (p. ex. Jaccard). Ils sont introduits dans le chapitre 2.

Ce type d'approche a notamment été utilisé par Ektefa *et al.* (2011). Les auteurs ont combiné différentes mesures de similarité pour comparer des enregistrements d'informations sur des restaurants en utilisant un seuil minimal de similarité. Ils ont démontré que la combinaison de plusieurs mesures surpasse l'utilisation individuelle des mesures.

1.1.2 Algorithme de classification

La seconde catégorie utilise un vecteur d'informations construit à partir de la comparaison des attributs des entités afin d'en permettre la classification (Christen, 2012a; Acheson *et al.*, 2019). Les deux classes ainsi formées sont « correspondants » (font référence à la même entreprise) et « non correspondants » (ne font pas référence à la même entreprise). Les algorithmes de classification doivent toutefois être entraînés afin d'être en mesure de classer correctement une comparaison entre deux entités. Il existe principalement deux paradigmes d'entraînement pour ce type d'algorithme : par apprentissage supervisé et par apprentissage non supervisé. Toutefois, nous nous intéressons ici uniquement à la première approche.

Apprentissage supervisé

L'apprentissage supervisé permet de prédire la classe d'une paire de comparaisons d'entreprises en se guidant à l'aide d'une étiquette-vérité (Christen, 2012a). C'est-à-dire que nous utilisons l'étiquette comme une forme d'enseignement permettant d'apprendre, à l'aide de données d'entraînement, à prédire la classe appropriée pour des données de test où la classe est absente (Shalev-Shwartz et Ben-David, 2014b). Parmi les algorithmes utilisant ce type d'apprentissage, on retrouve notamment les arbres de décision. L'un des avantages des algorithmes supervisés est qu'ils sont souvent plus faciles à entraîner. Toutefois, ils nécessitent souvent beaucoup de données annotées et parfois la création de variables explicatives pour obtenir de meilleurs résultats. Par exemple, la détection de doublons dans un répertoire de lieux géographiques a nécessité l'utilisation de 26 variables explicatives pour obtenir de bons résultats (Martins, 2011).

Ce type d'approche a notamment été utilisé par Bilenko et Mooney (2002). Les auteurs ont

réussi à utiliser une machine à vecteurs de support (MVS) pour prédire l'étiquette de la classe à partir des vecteurs d'informations entre deux enregistrements. Ces vecteurs d'informations correspondent à la combinaison de différents algorithmes de similarité adaptatifs ayant été appliqués sur plusieurs attributs. Les auteurs ont notamment utilisé l'algorithme adaptatif de Levenshtein qui apprend, selon le domaine d'application, le coût des opérations qui correspond à la différence entre deux chaînes de caractères pour ce type de données (Ristad et Yianilos, 1998). L'intuition étant que selon le type de données, la même différence entre deux chaînes de caractères ne représente pas nécessairement le même coût.

1.1.3 Sommaire de la section

En résumé, deux composantes sont nécessaires pour résoudre notre problématique : un algorithme de classification afin de déterminer la similarité entre les entreprises, et un algorithme de couplage afin de trouver, dans une source externe, l'entité ayant la similarité la plus élevée avec un risque commercial. Nous supposons donc qu'il existe au plus une seule entité pouvant être associée au même risque commercial. Cela signifie que le but de l'algorithme de couplage n'est pas de trouver les N doublons de ce dernier, mais plutôt de trouver uniquement l'entité la plus similaire. Le choix de cette approche restrictive est motivée par l'objectif d'extraire de l'information d'un risque commercial provenant d'une seule source externe plutôt que de déterminer les N doublons d'une même entité au sein de deux sources de données.

1.2 Solutions existantes

Puisque la détection de doublons est un problème récurrent dans les différents systèmes d'information, et particulièrement depuis l'avènement des bases de données massives (Naumann et Herschel, 2010; Govind *et al.*, 2019), diverses solutions logicielles existent déjà. On en retrouve deux types : celles dites « propriétaires », et celles en utilisation ouverte.

Solutions propriétaires

Les logiciels propriétaires, qui constituent la première catégorie, sont développés par des entreprises qui proposent des licences d'utilisation payantes, généralement dispendieuses. Ces options sont souvent contraignantes puisqu'elles nécessitent l'installation d'un logiciel. Par exemple, des solutions comme Data Ladder (2020), DQ Global (2020) et EGon (2020) permettent de détecter les doublons entre plusieurs sources de données. Ces solutions ne sont pas adaptées à un contexte d'automatisation des processus puisqu'elles nécessitent l'intervention humaine. Par ailleurs, elles sont peu flexibles dans un contexte de recherche puisqu'elles viennent déjà avec des solutions clé en main. Pour ces différentes raisons, peu de temps a été consacré à leur étude pour ce mémoire.

Solutions ouvertes

La seconde catégorie correspond à des logiciels ou des librairies développés par une communauté ou une entreprise et dont les licences sont souvent libres de droit. Ce type d'approche est plus intéressante que la précédente et mérite d'être étudiée plus en profondeur.

La première option, le logiciel libre *Febrl* développé par Christen (2008), permet de normaliser les données en plus de faire du couplage de données par approche probabiliste. Toutefois, ce logiciel est spécialisé dans le domaine d'application des données biomédicales et l'approche probabiliste n'est pas envisageable pour des données ayant uniquement deux attributs.

La seconde option, *DuDe*, est une librairie complète permettant la détection de doublons qui a été développée par Dra. La librairie contient notamment des extracteurs de données, des modules de prétraitement et des modules permettant la comparaison d'entités. Par contre, elle est disponible en Java et n'est pas supportée en Python, et elle ne propose aucune implémentation d'algorithmes d'apprentissage automatique. Il est donc nécessaire d'utiliser d'autres librairies d'apprentissage automatique en Java. De plus, la documentation de la librairie est difficilement accessible, vieille de plusieurs années, et il existe peu d'assistance sur les plateformes usuelles d'information (p. ex. *Stack Overflow*). L'utilisation de cette approche comporte donc son lot d'incertitudes sur la possibilité d'obtenir des réponses lorsqu'un débogage est nécessaire et sur la disponibilité de l'implantation d'algorithmes d'apprentissage automatique, ce qui la rend moins attrayante pour le développement d'algorithmes de classification plus avancés.

La dernière option intéressante, *py_entitymatching*, est une librairie en Python développée par Magellan Project (2020) qui offre divers outils permettant de développer une solution de détection de doublons entre deux sources de données. Elle propose notamment

— des outils permettant l'annotation,

— des méthodes de blocage des données permettant de limiter le nombre de comparaisons,

— la création d'attributs et

— des algorithmes d'apprentissage automatique supervisé.

Toutefois, les dépendances de la librairie ne sont pas maintenues à jour, et plusieurs de ses dernières sont vieilles d'au moins deux ans. La librairie utilise notamment les versions 0.23.2 de *Pandas* et 0.18.0 de *Scikit-learn*, qui sont maintenant respectivement 12 et 18 versions plus avancées. L'utilisation de librairies désuètes rend le développement vulnérable à des dysfonctionnements ayant été corrigés dans les versions subséquentes. En considérant que les mises à jour de *py_entitymatching* sont très irrégulières et peu fréquentes (cinq en trois ans), on peut s'attendre à ce que la librairie soit toujours de plus en plus désuète par rapport à ses dépendances. Cette désuétude fait en sorte qu'il est difficile d'utiliser certaines fonctionnalités de la librairie, et plusieurs manipulations complexes et ardues de gestion des dépendances seraient nécessaires pour y remédier.

De plus, certaines approches que nous avons utilisées ne sont pas facilement intégrables avec les outils de cette libraire, notamment la recherche en grille des paramètres des modèles d'apprentissage automatique et la génération d'attributs à l'aide de l'analyse en composantes principales.

Finalement, *py_entitymatching* utilise le tableau de données (*DataFrame*) de la libraire de visualisation et de manipulation de données *Pandas* comme structure de données. Cette structure de données a l'inconvénient de nécessiter une quantité importante de mémoire vive. Par exemple, les différentes manipulations nécessaires à faire sur les données ont été impossibles à effectuer, même en ayant à notre disposition près de 63 Go de mémoire vive.

En somme, puisque les dépendances désuètes et l'utilisation d'une structure de données peu efficace nous empêche d'effectuer les manipulations nécessaires à la résolution de la problématique, nous n'avons pas utilisé cette solution.

1.3 Sommaire du chapitre

Ce chapitre a présenté comment faire correspondre des entités entre différentes sources de données en utilisant un algorithme de détection de doublons. Deux catégories d'approche ont été présentées pour l'algorithme de détection de doublons : par comparaison d'attributs et par apprentissage automatique. De plus, les approches logicielles et les librairies déjà existantes ont aussi été abordées. Cependant, malgré la nature intéressante de la plupart de ses approches, nous nous concentrons sur la comparaison d'attributs à l'aide de mesures de similarité et sur les algorithmes de classification par apprentissage supervisé.

Le prochain chapitre présente la méthodologie utilisée dans ce mémoire pour préparer les sources de données ainsi que les méthodes employées pour déterminer la ressemblance entre les entités.

Chapitre 2

Méthodologie

Tel que discuté dans l'introduction, détecter des doublons entre deux sources de données non structurées est difficile pour trois raisons : la difficulté de représenter les données, la qualité de celles-ci et la complexité algorithmique. Ce chapitre présente comment cette problématique a été abordée dans ce mémoire. Nous y détaillons la méthodologie utilisée pour améliorer la qualité des données ainsi qu'une méthode pour limiter la complexité algorithmique. Nous terminons en décrivant les algorithmes utilisés pour représenter les données textuelles afin de pouvoir mesurer la similarité entre les entités.

2.1 Sources de données

Il a été discuté dans le chapitre précédent que pour tenter de résoudre la problématique de la détection de doublons, il est nécessaire d'avoir au moins deux sources de données, l'une correspondant à la liste de risques commerciaux que nous désirons coupler, et une correspondant à la source externe avec laquelle nous désirons recouper ces risques commerciaux. Cette section présente ces deux sources de données ainsi que les méthodes employées pour en améliorer la qualité et comment nous avons réduit le nombre d'opérations en utilisant un échantillon des données. Finalement, nous présentons les métriques utilisées pour évaluer les différentes approches.

2.1.1 Source de données propriétaire

Pour la première source, nous utilisons un jeu de données privé fourni par une entreprise partenaire. Cette source d'information contient le nom et l'adresse de 21 443 risques commerciaux canadiens recueillis par l'entreprise partenaire. Les noms et adresses des risques commerciaux sont bilingues puisque les données viennent exclusivement du Canada. Toutefois, 54,33 % des adresses postales sont du Québec. Cette caractéristique du jeu de données est intéressante, puisqu'elle nous permet de tirer profit du registre des entreprises du Québec (REQ) rendu public par le Registraire des entreprises (Registraire des entreprises du Québec, 2020b).

À cet effet, la source d'information privée a été filtrée afin de garder uniquement les entreprises basées au Québec, pour un total de 11 649. Nous avons gardé uniquement les adresses au Québec puisqu'il n'est pas possible pour les autres provinces d'obtenir gratuitement de l'information sur l'enregistrement d'une entreprise [1]. Le Tableau 2.1 présente l'information disponible sur une entreprise dans le jeu de données privé.

Secteur d'activité	MAN.2.4 (un code d'activité)
Activité	Clothing and Accessories
Nom	Identification sport inc.
Adresse	2233 ch du plateau N, Saguenay, Qc, G7B0G7
Latitude	48.349978
Longitude	-71.007167

Tableau 2.1 – Information disponible sur une entreprise du jeu de données privé

Deux étapes de nettoyage ont été effectuées sur le nom des risques commerciaux. La première consistait à retirer les noms qui correspondent à un nom « Québec inc. » (p. ex. 1234-5678 Québec inc.). En effet, au moment de l'immatriculation d'une société auprès du REQ, il est possible de demander une désignation numérique unique pour tenir lieu de nom (Registraire des entreprises du Québec, 2017). De par l'unicité de cette désignation, il y a peu d'intérêt à effectuer un couplage de ces entreprises parce qu'il est déjà certain d'emblée d'avoir un couplage exact. En effet, nos expérimentations préliminaires ont démontré un couplage presque parfait pour ces entreprises. Nous avons donc décidé de retirer ces entreprises puisqu'elles sont trop faciles à identifier par leur nom et ne présentent pas de défi pour nos travaux en similarité textuelle. Ainsi, lors de l'annotation du jeu de données, toutes les entreprises ayant uniquement un nom « Québec inc. », soit 1 094 entreprises, ont été mises de côté [2].

La seconde étape de nettoyage consistait à extraire le nom approprié à partir de plusieurs informations relatives au nom dans l'attribut correspondant parmi les 10 555 entreprises restantes. Dans certain cas, les noms étaient composés d'un nom « Québec inc. » et d'un second nom. Ces noms étaient écrits selon quatre patrons bien précis :

— o/a pour « *operating as* »,

— o/b pour « *on behalf* »,

— fasrs/f.a.s.r.s. pour « faisant affaire sous la raison sociale »,

— l'utilisation de la barre oblique (/).

Pour ces cas, une logique d'extraction du nom pertinent a été appliquée sur l'ensemble des noms du jeu de données filtré précédemment. Par exemple, pour le nom « 1234-5678 Québec inc. o/a Université Laval », l'extraction correspond à « Université Laval ».

1. Il existe une initiative canadienne permettant de faire la recherche d'une entreprise via le Registres d'entreprises au Canada, mais l'information n'est pas disponible gratuitement pour plusieurs provinces.

2. À noter que la même logique a aussi été appliquée sur les noms de risques commerciaux contenant une immatriculation hors Québec (p. ex. 123456 Canada inc.).

2.1.2 Source de données externe

Comme mentionné plus haut, la seconde source de données utilisée est le REQ (Registraire des entreprises du Québec, 2020b). Ce jeu de données est en fait l'historique des enregistrements des entreprises dans la province depuis la création du REQ le 1er janvier 1994 (Registraire des entreprises du Québec, 2020a). Ses quelques 2,3 millions d'enregistrements ne correspondent donc pas tous à des entités distinctes, puisqu'une entreprise pourrait s'être enregistrée à nouveau plutôt que de mettre à jour son dossier, créant ainsi une nouvelle entrée au registre. Par exemple, l'entreprise « Boulangerie St-Laurent » qui devient « Boulangerie St-Laurent enr. » demeure la même entreprise, mais on retrouvera deux enregistrements dans le jeu de données si elle fait un nouvel enregistrement plutôt que de mettre son dossier à jour.

Parmi les informations disponibles sur les entreprises listées, nous utilisons uniquement les différents noms francophones des entreprises, leurs noms dans d'autres langues, en l'occurrence l'anglais, et l'adresse. Le jeu de données du REQ a subi la même filtration que le jeu de données privé, soit le retrait des noms « Québec inc. » contenus dans les noms. Le Tableau 2.2 présente un extrait de l'information disponible dans le REQ à propos de l'Université Laval.

NEQ	1160043551
Nom	Université Laval
Adresse	2345 allée des Bibliothèques, Québec (Québec), G1V0A6
Secteur d'activité	Enseignement et recherche
Nombre de salariés	2500 à 4990
Date immatriculation	2001-04-18
Formation juridique	Personne morale sans but lucratif
Liste des administrateurs	Informations sur les 27 administrateurs
Liste des établissements	Nom, adresse et activité économiques des établissements
Liste des documents au dossier	Déclaration annuelle, courante et correction
Nom enregistré	Université Laval
Autres noms	Maison de percé, Observatoire St-Elzéard-de-Beauce

Tableau 2.2 – Un condensé de l'information disponible sur l'Université Laval dans le REQ

2.2 Évaluation

Pour rendre nos données de base utilisables dans les différentes approches dont nous voulons tester la précision, le rappel et l'exactitude, il nous a fallu créer une étiquette-vérité pour chaque paire risque commercial-entité. À cet effet, 1 706 risques commerciaux ont été annotées manuellement pour y ajouter le numéro d'entreprise du Québec correspondant dans le REQ. De ce nombre, 1 418 sont annotés positivement avec leur entité-vérité alors que le reste sont annotés négativement comme n'ayant pas d'entité-vérité, soit la valeur *None*. Puisque le nombre de données n'est pas équilibré entre les classes, il a été choisi de ne pas utiliser la F-mesure.

— Précision : nombre de risques commerciaux pertinents (positif) parmi les risques commerciaux couplés proposés, soit

$$\frac{\text{vrai positif}}{\text{vrai positif} \ + \ \text{faux positif}}.$$

— Rappel : nombre d'entreprises ayant été couplées avec leur entité-vérité parmi les exemples positifs, soit

$$\frac{\text{vrai positif}}{\text{vrai positif} \ + \ \text{faux négatif}}.$$

— Exactitude : nombre de bons couplages parmi l'ensemble des couplages effectués (positif et négatifs), soit

$$\frac{\text{vrai positif} \ + \ \text{vrai négatif}}{\text{vrai positif} \ + \ \text{vrai négatif} \ + \ \text{faux positif} \ + \ \text{faux négatif}}.$$

Les exemples qui ont été couplés avec la mauvaise entité du REQ sont considérés comme des faux négatifs pour le calcul des métriques. Par exemple, si l'entreprise A a été associée avec l'entité C alors que l'entité-vérité est l'entité B, alors il s'agit d'un faux négatif.

Finalement, puisque l'un de nos objectifs est de réduire le nombre de questions du questionnaire client et qu'il est possible de valider ces réponses avec le client, nous désirons donc mettre en relation le plus d'entreprises possible avec une bonne précision. Cela signifie que nous devons maximiser le rappel tout en maintenant une bonne précision et une bonne exactitude.

2.3 Indexation

Étant donné que la source externe est volumineuse, et tel que nous en avons discuté plus tôt, une approche naïve pour coupler les 1 706 risques commerciaux serait de comparer l'ensemble de ceux-ci avec les 2,3 millions d'entités du REQ. En utilisant cette approche, nous effectuerions un total de 1 706 $\times$ 2,3 millions de comparaisons. Même en considérant que chaque opération s'effectue en 0,001 seconde, environ 45 jours seraient nécessaires pour effectuer l'ensemble des opérations. et ce pour une seule méthode. Puisque l'un des objectifs est d'assister la saisie d'informations dans un questionnaire, il est évident qu'une telle durée d'attente n'a absolument aucun sens. Il est donc nécessaire de comparer le risque commercial uniquement avec les entreprises ayant des caractéristiques du nom ou de l'adresse en commun avec lui. Par exemple, le nom « Boulangerie St-Laurent » contient plusieurs mots en commun avec « Boulangerie St-Laurent enr. », mais n'en partage aucun avec « Université Laval », il est donc inutile de le comparer avec ce dernier. Autrement dit, parmi les 2,3 millions de comparaisons, de nombreuses seraient inutiles. Pour cette raison, nous avons utilisé un index textuel pour y effectuer des requêtes en utilisant Whoosh (Chaput, 2017). La Figure 2.1 présente un exemple d'utilisation de cet index pour le couplage de l'entreprise « Boulangerie St-Laurent ».

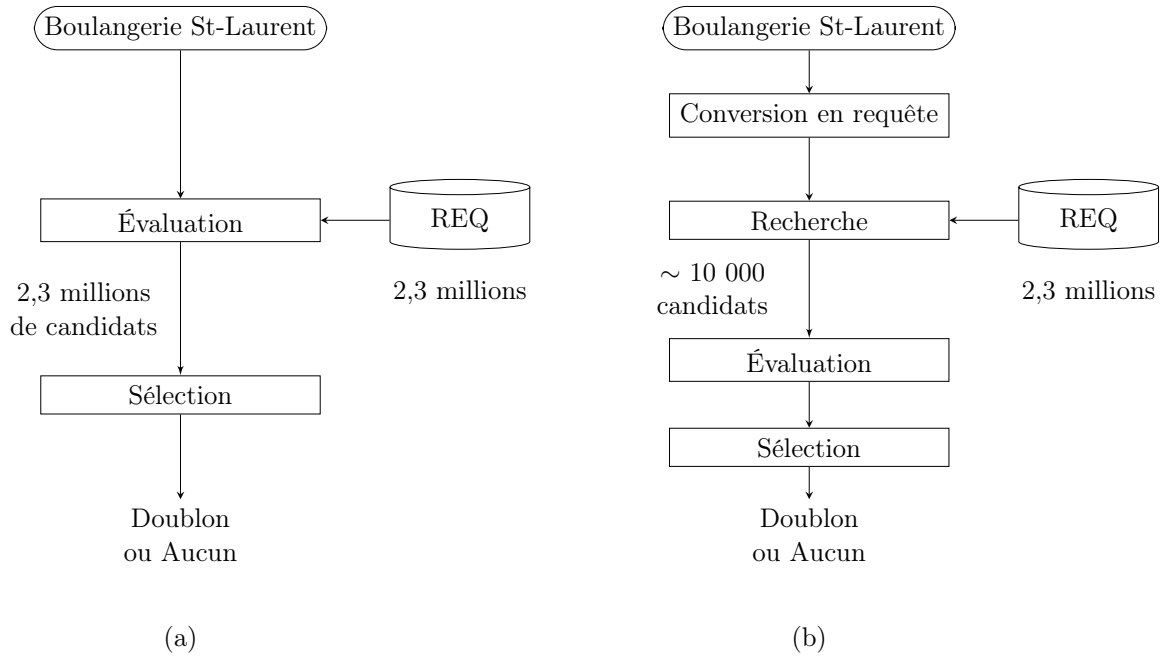

FIGURE 2.1 – Étapes du processus de détection de doublons (a) Comparaison de l'entreprise avec toutes les entités du REQ (b) Filtrage des entités du REQ avec un système de recherche d'information

Cet index est composé des noms et de l'adresse de l'ensemble des entreprises du REQ. Puisqu'une entité peut avoir jusqu'à 43 774 noms différents, mais que 98,37 % d'entre elles en ont au plus six, nous n'avons utilisé au plus les six premiers noms d'une entreprises.

2.3.1 Les requêtes

Pour effectuer la recherche des entreprises intéressantes à comparer avec le risque commercial, nous avons utilisé deux requêtes pour chacun des attributs nom et adresse. Individuellement, les requêtes ne permettent pas d'obtenir un nombre suffisant d'entités pertinentes à comparer avec le risque commercial, mais utilisées conjointement, elles permettent d'en obtenir entre 9 000 et 11 000. Cela représente une diminution significative par rapport aux 2,3 millions de comparaisons à l'origine.

Nous n'avons pas utilisé de requêtes combinées à la fois sur le nom et l'adresse puisque cela ne permettait pas d'obtenir plus d'entités pertinentes. Nous n'avons pas utilisé de requêtes qui combinent à la fois le nom et l'adresse d'une entreprise puisque cela ne permet pas de repérer de nouvelles entités pertinentes qui seraient omises par des requêtes individuelles.

Requêtes sur le nom

La première requête sur le nom consiste à faire correspondre un minimum de trois mots à travers les six noms présents dans l'entité. Cette requête permet d'obtenir en moyenne

8 100 entités à comparer, et l'entité-vérité s'y retrouve dans près de 83 % des cas.

La seconde requête consiste en la sélection des 1 000 premiers résultats de l'algorithme de tri BM25F (Robertson et Sparck Jones, 1988) en utilisant les mêmes mots que la première requête. Cet algorithme attribue un score par pondération de la similarité entre les entités en fonction de la fréquence d'apparition des mots recherchés dans chacune d'entre elles. Nous avons utilisé cet algorithme puisqu'il s'agit de celui par défaut dans la librairie Whoosh (Chaput, 2017) et qu'il permet d'obtenir de bons résultats de tri (Robertson, 2009). De plus, puisque nous utilisons par la suite d'autres mesures de similarité, le classement des documents obtenus par l'algorithme de tri n'a pas d'impact sur le choix final. L'étape de tri sert uniquement à filtrer les candidats potentiels. Cette requête permet d'obtenir en moyenne 900 documents supplémentaires à comparer, une fois les doublons avec les résultats de la première requête retirés.

Ces deux requêtes successives permettent finalement d'obtenir en moyenne 9 000 entités avec lesquelles comparer le risque commercial. L'entité-vérité se retrouve dans cet échantillon dans près de 93 % des cas. Cela signifie que la plus haute valeur de rappel possible ne pourra excéder 93 %.

Requêtes sur l'adresse

Pour l'adresse, nous avons utilisé les mêmes deux requêtes que pour le nom mais en utilisant l'unique adresse des entreprises. Autrement dit, nous cherchons soit une entité ayant au moins trois mots similaires dans l'adresse ou les 1 000 premiers documents triés par l'algorithme BM25F. Ces deux requêtes permettent d'obtenir en moyenne 11 000 documents parmi lesquels nous retrouvons l'entité-vérité dans 90 % des cas.

2.4 Correspondance par estimation de similarité

Pour effectuer le couplage d'un risque commercial avec une entité du REQ, il est nécessaire de quantifier le degré de ressemblance entre ces deux éléments. Puisque notre objectif est de trouver l'entité ayant la ressemblance la plus élevée avec le risque commercial, nous transformons cette ressemblance en un score normalisé entre 0 et 1 afin qu'elle soit mieux représentable pour un utilisateur. Par exemple, une similarité de « 0,85 » (soit 85 %) est plus facilement compréhensible qu'une similarité de « 2,5 ». Ainsi, 0 correspond à une dissemblance entre les deux éléments, et 1 à une ressemblance parfaite.

Une première approche consiste à utiliser un algorithme de similarité afin de pouvoir déterminer le niveau de ressemblance entre deux chaînes de caractères (A et B). La littérature propose plusieurs algorithmes permettant de calculer cette similarité, notamment les algorithmes de Jaccard (Jaccard, 1901), de Jaro (Jaro, 1989) ou de Jaro-Winkler (Winkler, 1990). Toutefois, Cohen *et al.* (2003b) ont complété une analyse exhaustive de l'efficacité de ces nombreux algo-

rithmes sur l'association du nom. Ils ont aussi séparé ces derniers en trois types : par distance d'édition, à jetons et hybride. Nous nous sommes appuyé sur cette analyse pour sélectionner les algorithmes de similarité qui font partie de notre étude. Ceux-ci sont décrits dans les sous-sections suivantes.

2.4.1 Algorithme par distance d'édition

Cette catégorie d'algorithme mesure la similarité entre deux chaînes de caractères en s'intéressant au nombre d'opérations nécessaires pour convertir une chaîne de caractères A en une chaîne B. Nous avons retenu cinq algorithmes de ce type : la comparaison par chaîne de caractères (CCC), de Levenshtein (Yujian et Bo, 2007), de Jaro (Jaro, 1989), de Jaro-Winkler (Winkler, 1990) et l'algorithme de la plus longue sous-séquence commune (PLSC) (Hirschberg, 1977).

La comparaison par chaîne de caractères est une mesure binaire de comparaison entre les chaînes de caractères A et B. Cette approche détermine si la chaîne de caractères A est parfaitement identique à la chaîne de caractères B. C'est-à-dire que si la séquence de caractères de A est identique à celle de B, alors la similarité est de 1, et de 0 autrement. Ce qui correspond à l'Équation 2.1.

$$\mathrm{CCC(A,\,B)} = \begin{cases} 1 & \text{si } \mathrm{A}_i = \mathrm{B}_j \ \forall i, j \\ 0 & \text{autrement} \end{cases} \tag{2.1}$$

La distance de Levenshtein est le plus petit nombre d'opérations à coût unitaire de suppression, d'insertion ou de substitution nécessaires pour convertir la chaîne de caractères A en B. Pour normaliser la distance, on divise la plus petite distance par la longueur de la plus longue chaîne de caractères. On obtient finalement la similarité en effectuant $1 -$ distance, ce qui correspond à l'Équation 2.2.

$$\text{Levenshtein normalisé(A, B)} = 1 - \frac{\mathrm{Lev}_{\mathrm{A,\,B}}(i, j)}{\max\left(|\mathrm{A}|, |\mathrm{B}|\right)} \tag{2.2}$$

La fonction $\mathrm{Lev}_{\mathrm{A,\,B}}(i, j)$ correspond à l'Équation 2.3.

$$\mathrm{Lev}_{\mathrm{A,\,B}}(i, j) = \begin{cases} \max(i, j) & \text{si } \min(i, j) = 0 \\ \min \begin{cases} \mathrm{Lev}_{\mathrm{A,\,B}}(i-1, j) + 1 \text{ (suppression)} \\ \mathrm{Lev}_{\mathrm{A,\,B}}(i, j-1) + 1 \text{ (insertion)} \\ \mathrm{Lev}_{\mathrm{A,\,B}}(i-1, j-1) + 1_{(\mathrm{A}_i \neq \mathrm{B}_j)} \text{ (substitution)} \end{cases} & \text{autrement} \end{cases} \tag{2.3}$$

Où i est le i-ième caractère de A, j est le j-ième caractère de B, et $1_{(\mathrm{A}_i \neq \mathrm{B}_j)}$ est une fonction indicatrice égale à 1 si $\mathrm{A}_i \neq \mathrm{B}_j$ (c.-à-d. le caractère i est différent du caractère j).

Les mesures de similarités de Jaro et de Jaro-Winkler sont deux algorithmes fortement reliés, le second étant une variante du premier. On les utilise principalement pour des chaînes de caractères relativement courtes telles que des prénoms ou des noms de famille.

La similarité de Jaro équivaut à la moyenne du nombre de caractères correspondants entre les chaînes A et B (m) sur leurs longueurs respectives et le nombre de caractères non correspondants (t) sur m. On dit que deux caractères correspondent s'ils sont identiques et sont à une distance l'un de l'autre d'au plus

$$\left\lfloor \frac{\max\left(|\mathrm{A}|, |\mathrm{B}|\right)}{2} \right\rfloor - 1.$$

On obtient ainsi l'Équation 2.4 pour illustrer la similarité de Jaro.

$$\mathrm{Jaro(A,\ B)} = \begin{cases} 0 & \text{si } m = 0 \\ \frac{1}{3}\left(\frac{m}{|\mathrm{A}|} + \frac{m}{|\mathrm{B}|} + \frac{m-t}{m}\right) & \text{autrement} \end{cases} \tag{2.4}$$

Par exemple, le caractère « a » dans « abcde » est correspondant au même caractère dans « zacde ». De plus, les caractères « cde » sont aussi correspondants, ce qui signifie que $m = 4$ et $t = 1$. La valeur de la similarité correspond donc à

$$\frac{1}{3}\left(\frac{4}{5} + \frac{4}{5} + \frac{4-1}{4}\right) = 0{,}78.$$

L'algorithme de Jaro-Winkler accorde une similarité plus élevée aux chaînes de caractères qui partagent un préfixe (P) similaire d'une longueur maximale de quatre caractères, et qui est calculée à partir de l'algorithme de Jaro. Ce qui correspond à l'Équation 2.5.

$$\mathrm{Jaro\text{-}Winkler(A,\ B)} = \mathrm{Jaro(A,\ B)} + \frac{\min\left(P, 4\right)}{10} \times \left(1 - \mathrm{Jaro(A,\ B)}\right) \tag{2.5}$$

Le dernier algorithme, la PLSC, calcule la plus longue sous-séquence identique entre A et B en débutant à n'importe quelle position dans les chaînes de caractères (Bergroth *et al.*, 2000). Pour normaliser cette similarité, on divise la longueur de la plus longue sous-séquence par la longueur de la plus longue chaîne de caractères. Ce qui correspond à l'Équation 2.7.

$$\mathrm{PLSC\ normalisé(A,\ B)} = \frac{\mathrm{PLSC}_{\mathrm{A,\ B}}(i, j)}{\max\left(|\mathrm{A}|, |\mathrm{B}|\right)} \tag{2.6}$$

La fonction $\mathrm{PLSC}_{\mathrm{A,\,B}}(i,j)$ correspond à l'Équation 2.7.

$$\mathrm{PLSC}_{\mathrm{A,\,B}}(i,j) = \begin{cases} 0 & \text{si } \min(|\mathrm{A}|,|\mathrm{B}|) = 0 \\ \mathrm{PLSC}_{\mathrm{A,\,B}}(i-1,j-1)+1 & \text{si } \mathrm{A}_i = \mathrm{B}_j \\ \max\left(\mathrm{PLSC}_{\mathrm{A,\,B}}(i,j-1),\, \mathrm{PLSC}_{\mathrm{A,\,B}}(i-1,j)\right) & \text{si } \mathrm{A}_i \neq \mathrm{B}_j \end{cases} \tag{2.7}$$

Où i est le i-ième caractère de A et j est le j-ième caractère de B.

Pour illustrer la mesure de la PLSC, la similarité entre les chaînes de caractères « Université Laval » et « Université de Montréal » correspond à

$$\frac{|\text{Université}|}{|\text{Université de Montréal}|} = \frac{11}{22} = \mathbf{0{,}5}$$

alors qu'elle sera de 0,91, 0,61 et 0,61 pour Levenshtein, Jaro et Jaro-Winkler respectivement.

2.4.2 Algorithme à jetons

Cette catégorie d'algorithme mesure la similarité entre deux chaînes de caractères en comparant les « jetons » de mots présents dans les chaînes de caractères. Par exemple, « Université Laval » contient deux jetons de mot, « Université » et « Laval ». Nous avons retenu quatre algorithmes dans cette catégorie : de Jaccard (Jaccard, 1901), du cosinus (Singhal, 2001), du coefficient de Szymkiewick-Simpson (CSS) (Sempson, 1947) et le *Measuring Agreement on Set-valued Items* (MASI) (Passonneau *et al.*, 2006).

L'algorithme de Jaccard mesure la similarité entre des chaînes de caractères en calculant le ratio du nombre de jetons de leur intersections par rapport au nombre de jetons de leur union. Ce qui correspond à l'Équation 2.8.

$$\mathrm{Jaccard}(\mathrm{A},\,\mathrm{B}) = \begin{cases} 0 & \text{si } |\mathrm{A}\cap\mathrm{B}| = 0 \text{ ou } |\mathrm{A}\cup\mathrm{B}| = 0 \\ \frac{|\mathrm{A}\cap\mathrm{B}|}{|\mathrm{A}\cup\mathrm{B}|} & \text{autrement} \end{cases} \tag{2.8}$$

Le cosinus mesure la similarité en calculant le produit scalaire entre les vecteurs des chaînes de caractères. On détermine le vecteur d'une chaîne de caractères en utilisant la méthode *TF-IDF* [3] (Sammut et Webb, 2010) qui pondère les jetons de mot selon leur importance. Ce qui correspond à l'Équation 2.9.

$$\mathrm{cosinus}(\mathrm{A},\,\mathrm{B}) = \sum_{w\in\mathrm{A}\cap\mathrm{B}} V(w,A)\times V(w,B) \tag{2.9}$$

3. De l'anglais *term frequency-inverse document frequency.*

où $V(w, s_i)$ correspond à

$$V(w, s_i) = \frac{V'(w, s_i)}{\sqrt{\sum_w V'(w, s_i)^2}}$$

et $V'(w, s_i)$ correspond à

$$V'(w, s_i) = \log(\mathrm{TF}_{w,s_i} + 1) \times \log(\mathrm{IDF}_w)$$

avec IDF qui correspond à la fréquence inverse de document, soit la mesure de l'importance du terme dans l'ensemble du corpus en accordant un poids plus élevés aux termes les moins fréquents.

Le CSS est semblable à Jaccard, mais plutôt que d'utiliser le nombre de jetons de mot de l'intersection entre les chaînes de caractères pour effectuer le ratio, on utilise le nombre de jetons de mot dans la chaîne de caractères qui en contient le moins. Ce qui correspond à l'Équation 2.10.

$$\mathrm{CSS(A,\ B)} = \begin{cases} 0 & \text{si } |\mathrm{A} \cap \mathrm{B}| = 0 \text{ ou } \min(|\mathrm{A}|, |\mathrm{B}|) = 0 \\ \frac{|\mathrm{A} \cap \mathrm{B}|}{\min(|\mathrm{A}|, |\mathrm{B}|)} & \text{autrement} \end{cases} \tag{2.10}$$

Le MASI correspond au résultat de Jaccard pondérée par un facteur M déterminé par l'intersection des ensembles de jetons de mot des deux chaînes de caractères. Ce qui correspond à l'Équation 2.11.

$$\mathrm{MASI(A,\ B)} = \mathrm{Jaccard(A,\ B)} \times \mathrm{M(A,\ B)} \tag{2.11}$$

où M(A, B) correspond à

$$\mathrm{M(A,\ B)} = \begin{cases} 1 & \text{si } s_\mathrm{A} = s_\mathrm{B} \\ \frac{2}{3} & \text{si } s_\mathrm{A} \subset s_\mathrm{B} \vee s_\mathrm{B} \subset s_\mathrm{A} \\ \frac{1}{3} & \text{si } s_\mathrm{A} \cap s_\mathrm{B} \neq 0 \wedge s_\mathrm{A} \neq \emptyset \wedge s_\mathrm{B} \neq \emptyset \\ 0 & \text{autrement} \end{cases} \tag{2.12}$$

avec s_A et s_B qui correspondent à l'ensemble des quatre premiers jetons de caractères de leurs chaînes de caractères respectives.

Pour illustrer la mesure de Jaccard, la similarité entre les chaînes de caractères « Université Laval » et « Université de Montréal » correspond à

$$\frac{|\,\{\text{Université}\}\,|}{|\,\{\text{Université, de, Montréal, Laval}\}\,|} = \frac{1}{4} = \mathbf{0{,}25}$$

alors qu'elle est de 0,41, 0,5 et 0,08 pour le cosinus, le CSS et le MASI respectivement.

2.4.3 Monge Elkan

Cet algorithme par récursion mesure la similarité entre deux chaînes de caractères en combinant les approches par distance d'édition et par jetons (Cohen *et al.*, 2003a). Il fonctionne en pondérant la similarité normalisée de Damerau-Levenshtein (Brill et Moore, 2000) entre chaque jeton de mot des chaînes de caractères. Cette méthode est très similaire à celle de Levenshtein sauf qu'on y ajoute l'opération de transposition entre deux caractères (Bard, 2007). Pour normaliser la similarité de Monge Elkan, on divise celle-ci par la longueur de la plus longue chaîne de caractères, ce qui correspond à l'Équation 2.13.

$$\text{Monge Elkan normalisé}(A, B) = \frac{\text{Monge Elkan(A, B)}}{\max\left(|\text{A}|, |\text{B}|\right)} \tag{2.13}$$

où la fonction Monge Elkan(A, B) correspond à l'Équation 2.14.

$$\text{Monge Elkan(A, B)} = \frac{1}{K}\sum_{i=1}^{K}\max_{j=1}^{L}\left(\text{Damerau-Levenshtein}\left(\text{A}_i, \text{B}_j\right)\right) \tag{2.14}$$

avec K et L correspondant au nombre de mots dans les chaîne de caractères A et B respectivement. Cela signifie que A_i correspond au i-ième mot de A, et idem pour B avec l'indice j.

2.5 Correspondance par apprentissage automatique

Une seconde approche pour représenter la ressemblance entre une entreprise d'une base de donnée et une autre du REQ est l'utilisation d'algorithmes d'apprentissage automatique. Ces derniers reposent sur l'intuition qu'une entité (entreprise) est en fait une structure composée d'attributs sur lesquels il est possible d'utiliser une fonction de décision binaire, par exemple un arbre de décision, pour déterminer la correspondance entre deux entités. Divers algorithmes de liaison de données permettant de déterminer une similarité sont proposés dans la littérature (Rosenblatt, 1958; Safavian et Landgrebe, 1991; Hearst, 1998; Breiman, 2001), mais ces approches nécessitent l'utilisation d'un vecteur d'attributs numérique. Il est donc nécessaire de convertir en variables catégoriques ou numériques les attributs textuels des entités, soit le nom et l'adresse dans notre cas.

Cerda *et al.* (2018) ont analysé les diverses méthodes d'encodage des variables catégoriques à partir de données textuelles. La méthode la plus simple est l'utilisation d'une variable indicatrice qui aurait la valeur 1 lorsqu'une information est présente, et 0 autrement (p. ex. $1_{\text{nom = université laval}}$). Dans notre cas, cela revient à avoir une variable indicatrice pour chaque élément, soit un encodage 1 parmi n. En utilisant cette approche par matrice creuse, on

augmente grandement la dimensionnalité de la problématique et son empreinte informatique. Considérant que l'empreinte informatique doit être limitée dans notre cas, nous avons rejeté cette approche.

Parmi les autres solutions, celle qui semble être la plus prometteuse est celle de Martins (2011). Elle consiste à utiliser des algorithmes de similarité pour construire un vecteur d'informations. Par exemple, pour déterminer la similarité entre « Université Laval » et « Université de Montréal », on utilise des algorithmes de similarité (p. ex. Jaccard, PLSC, CCC) entre les noms des entités pour créer le vecteur d'informations. Par exemple, le vecteur d'informations composé des algorithmes de Jaccard, de la PLSC et de la CCC correspond à $[0{,}25; 0{,}5; 0]$. Cette approche pour convertir les attributs en un vecteur d'informations sera décrit plus en profondeur au chapitre 5.

Enfin, pour déterminer la correspondance entre deux entités à partir du vecteur d'informations, nous utilisons un algorithme de classification qui permet de vérifier si ces entités font référence à la même entreprise. Nous avons retenus deux algorithmes d'apprentissage automatique de types interprétable, et un de type non-interprétable, parmi tous ceux évalués. Le choix d'au moins une méthode de type interprétable permet d'évaluer si le gain en performance souvent associé aux méthodes non interprétables justifie l'utilisation exclusive de ces dernières.

2.5.1 Algorithmes de classification interprétables

Les deux algorithmes de classification interprétables sont la régression logistique et les forêts aléatoires d'arbres de décision. Nous avons sélectionné ces approches puisqu'elles sont rapides à entrainer, qu'elles ne nécessitent pas beaucoup d'exemples d'entrainement et qu'elles sont faciles à utiliser en pratique (Jurafsky et Martin, 2009b; Shalev-Shwartz et Ben-David, 2014a).

Régression logistique

La régression logistique binaire permet de modéliser la probabilité qu'une observation X soit de la classe 1 (même entité) ou 0 (entité différente). Pour modéliser cette probabilité, on utilise un vecteur d'informations (x_i) que l'on multiplie par un vecteur de poids (β_i). Ce vecteur de poids est appris à partir des exemples d'entraînement (Jurafsky et Martin, 2009b). On peut représenter la probabilité qu'une observation X appartienne à la classe 1 par l'Équation 2.15.

$$P(X = 1) = \frac{1}{1 + e^{-(\beta_1 \times x_1 + \ldots + \beta_n \times x_n)}} \tag{2.15}$$

Il est attendu qu'une observation X ayant un vecteur d'informations composé uniquement de zéros (c.-à-d. $x_i = 0 \ \forall \ i$) ait une similarité nulle. La probabilité d'être dans la classe 1 doit elle aussi être de 0, c'est pourquoi la valeur du paramètre d'ordonnée à l'origine (β_0) est nulle, sinon un vecteur de zéros pourrait avoir une probabilité non nulle.

Forêt aléatoire

Pour introduire la notion de forêt aléatoire, il faut d'abord parler de l'arbre de décision (bre), soit un algorithme dont le but est de prédire la classe Y d'un élément X en parcourant un arbre de son nœud racine jusqu'à une feuille finale associée à l'une des classes 0 ou 1 (Shalev-Shwartz et Ben-David, 2014a). Un arbre binaire est ainsi composé d'au moins un nœud et de deux feuilles où chaque nœud est un test sur un attribut permettant de faire progresser la classification de l'exemple. Un algorithme d'apprentissage construit un arbre de classification à partir du sommet en choisissant à chaque étape l'attribut qui réalise le meilleur gain de partition parmi l'ensemble des objets. Pour déterminer cet attribut sur un nœud, ce dernier tente de maximiser un critère donné. Dans le cas d'un arbre de classification binaire, deux critères sont utilisés : l'indice de Gini et le gain d'information (Shalev-Shwartz et Ben-David, 2014a). Raileanu et Stoffel (2004) ont démontré que pour un arbre de classification, l'utilisation de l'un ou de l'autre de ces critères permet d'obtenir des résultats très similaires. Par contre, comme le gain d'information utilise la fonction logarithmique, il est computationnellement plus efficace d'utiliser l'indice de Gini. La Figure 2.2 présente un exemple d'arbre binaire composé de deux nœuds qui permet de vérifier si deux entités sont la même à partir de leurs noms.

Le premier nœud se lit comme suit : si la similarité de Jaccard entre les noms des entités A et B est plus grande ou égale à 0,75, alors il s'agit de la même entité (bleu), sinon, on passe au second nœud.

Le second nœud se lit comme suit : si la similarité de Jaro-Winkler entre les noms est plus grande ou égale à 0,76, alors il s'agit de la même entité, sinon, les entités sont différentes (orange).

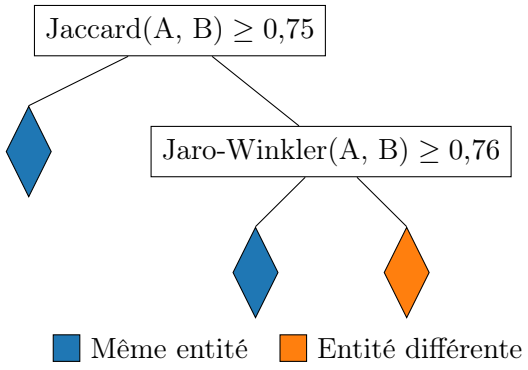

FIGURE 2.2 – Exemple d'arbre de classification binaire à deux nœuds sur l'attribut du nom

Toutefois, l'un des problèmes avec les arbres de décision est qu'il est souvent nécessaire de les élaguer afin de limiter le nombre de nœuds et de feuilles et ainsi minimiser le sur-apprentissage (Shalev-Shwartz et Ben-David, 2014a). Un autre moyen pour réduire le risque de sur-apprentissage est l'utilisation d'un ensemble d'arbres de décision binaire (Breiman, 2001)

par un algorithme d'entraînement sur des sous-ensembles de données légèrement différents les uns des autres. La collection d'arbres entraînés ainsi obtenue permet de prédire la classe de l'observation X en utilisant un vote de majorité, c'est-à-dire que la classe ayant reçu le plus de votes de la part des N arbres sera la classe prédite par la forêt aléatoire (Shalev-Shwartz et Ben-David, 2014a). Probst et Boulesteix (2017) ont démontré empiriquement que le nombre optimal d'arbres N semble être le nombre d'arbres plus élevé et le plus computationnellement raisonnable possible.

Puisque les arbres binaires souffrent du problème de sur-apprentissage, nous n'avons pas retenu ce type algorithme. Nous avons cependant utilisé une forêt aléatoire d'arbres de classification.

2.5.2 Algorithme de classification non interprétable

Le seul algorithme de classification de données non interprétable que nous avons étudié est le perceptron multicouche (Rosenblatt, 1958). Nous avons sélectionné cette approche puisque celle-ci est relativement rapide à entrainer pour un algorithme d'apprentissage profond, qu'elle nécessite un nombre raisonnable d'exemples d'entrainement et qu'elle démontre de bonnes propriétés de généralisation (Shalev-Shwartz et Ben-David, 2014c).

Le perceptron est un réseau de neurones pleinement connectés constitué d'un ensemble de perceptrons constituant une couche organisée et où chacun utilise une fonction d'activation, par exemple tanh, pour effectuer une classification binaire. Chaque neurone reçoit ainsi un signal d'entrée, qui l'activera ou non. Ce signal sera par la suite multiplié par un poids de sortie et sera ensuite passé à la prochaine couche. Cette étape sera répétée pour chaque couche jusqu'à la dernière. On obtient ensuite la prédiction de la classe en appliquant la fonction softmax sur le vecteur de poids de sortie de la dernière couche. Lors de l'entraînement, les poids des liens entre les couches sont optimisés afin d'obtenir la meilleure prédiction pour l'ensemble des observations. La Figure 2.3 présente un exemple de perceptron multicouche binaire avec L couches cachées et une couche de sortie binaire.

2.6 Sommaire du chapitre

Dans ce chapitre, nous avons présenté les deux sources de données utilisées dans ce mémoire, soit un jeu de données propriétaire de risques commerciaux et le registre des entreprises du Québec. Nous avons aussi présenté la méthodologie utilisée pour nettoyer les données. Le recours à un index textuel a permis de réduire la complexité algorithmique de la recherche d'entités potentiellement similaires. De plus, les métriques utilisées pour comparer les algorithmes ont été décrites.

Nous avons également présenté les deux familles d'algorithmes permettant de mesurer la ressemblance entre deux chaînes de caractères que nous utiliserons.

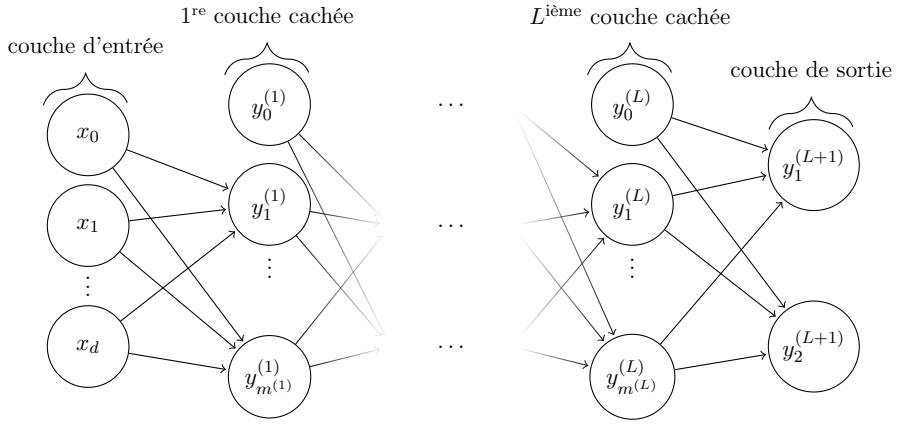

FIGURE 2.3 – Représentation d'un perceptron multicouche de classification binaire de $(L+1)$ couches avec un vecteur d'entrée de dimension d et deux unités de sortie où la $L^{\text{ième}}$ couche cachée contient $m^{(l)}$ neurones

La première famille est constituée des algorithmes de similarité qui sont utilisés spécifiquement pour déterminer la ressemblance entre deux chaînes de caractères. Ils n'utilisent pas tous la même unité de texte pour déterminer les éléments semblables. Certains utilisent la distance d'édition entre les caractères des deux éléments comparés alors que d'autres comparent les jetons de mot. Ces algorithmes ont l'avantage d'être très simples à utiliser, mais ont parfois une empreinte de calcul importante.

La seconde famille est quand à elle composée des algorithmes d'apprentissage automatique qui sont utilisés pour déterminer la ressemblance entre deux entités à partir de plusieurs attributs. Ils requièrent 'un vecteur d'informations composé des résultats générés par les algorithmes de similarités. Sa constitution est abordée plus loin dans ce mémoire.

Dans le prochain chapitre, nous présentons les résultats de l'utilisation des algorithmes de similarité pour le couplage du nom.

Chapitre 3

Couplage de noms d'entreprises par mesure de similarité

Dans ce chapitre, nous présentons les résultats du couplage du nom des risques commerciaux en utilisant les mesures de similarité présentées à la section 2.4. De plus, nous proposons une approche permettant d'améliorer les résultats observés.

3.1 Couplage du nom

L'algorithme 1 décrit comment effectuer le couplage d'une variante du nom d'un risque commercial avec la plus proche entité du registre des entreprises du Québec (REQ). Cet algorithme repose sur le calcul de la similarité (ligne 7) entre la variante du nom du risque commercial et l'ensemble des variantes des différents noms d'une entité du REQ. L'entité dont le nom d'entreprise a la plus haute similarité avec le nom à associer est considérée comme la plus similaire à cette dernière.

Algorithme 1 Couplage du nom d'un risque commercial

```
1:  procédure MEILLEUR COUPLAGE(risque)
2:      meilleure_similarité ← 0
3:      meilleur_couplage ← aucun
4:      entités_candidates ← requête_index(risque.nom)
5:      pour chaque entité_REQ dans entités_candidates faire
6:          nom ← entité_REQ.nom
7:          similarité ← calculer_similarité(risque.nom, nom)
8:          si similarité > meilleure_similarité alors
9:              meilleure_similarité ← similarité
10:             meilleur_couplage ← entité_REQ
11:     retourne meilleur_couplage
```

Variantes des noms

Pour effectuer le couplage, nous utilisons soit la variante du nom normalisé, soit celle du nom sans mots outils.

La normalisation d'un nom d'entité consiste en :

— la mise en minuscules,

— au retrait des espaces superflus en début et en fin de nom,

— au retrait des accents et des caractères spéciaux (p. ex. ç, â, é, ').

Par exemple, lorsqu'on utilise cette procédure de normalisation sur le nom « L'Université Laval », on obtient « l'universite laval ».

Dans la seconde variante, on applique une étape de retrait des mots outils appliquée après la normalisation. Les mots outils sont des mots très communs et qui donnent peu de signification dans un texte (p. ex. le, la, l') (Jurafsky et Martin, 2009c). Par exemple, « l'universite laval » devient « universite laval ».

3.2 Couplage des exemples positifs

L'objectif de cette section est d'analyser les résultats du couplage des variantes du nom avec les mesures de similarité, soit les algorithmes de la comparaison par chaîne de caractères (CCC), de Levenshtein, de Jaro, de Jaro-Winkler, de la plus longue sous-séquence commune (PLSC), de Jaccard, du cosinus, du coefficient de Szymkiewick-Simpson (CSS), le MASI et de Monge Elkan.

Pour cette expérience, nous utilisons uniquement les 1 418 risques commerciaux ayant une entité-vérité dans le REQ afin de permettre la sélection des algorithmes pertinents.

3.2.1 Approches par distance d'édition

Le Tableau 3.1 présente les résultats des approches par distance d'édition. On observe que les meilleurs résultats sont obtenus avec les algorithmes de Jaro, de Jaro-Winkler et de la PLSC. Ces derniers permettent d'obtenir une exactitude entre 63 % et 65 % selon la variante du nom. Les résultats les plus bas sont ceux de la CCC avec une exactitude autour de 42 %.

De plus, malgré la nature restrictive de la CCC, qui accorde une similarité de 1 uniquement lorsque les deux noms sont parfaitement identiques, ses résultats sont relativement bons. En utilisant le nom sans mots outils, 44,15 % des risques commerciaux sont couplés avec leur entité-vérité comparativement à 41,47 % avec le nom normalisé. Cela signifie que pour environ 41 % des risques commerciaux, il est possible de trouver l'entité-vérité portant exactement le même nom en effectuant une normalisation relativement simple. Ce nombre augmente légèrement, soit d'environ 3 %, lorsqu'on utilise une normalisation et le retrait des mots outils. Cela

signifie que très peu de risques commerciaux ont des noms étant à quelques mots outils près de leur entité-vérité.

	(%)	CCC	Levenshtein	Jaro	Jaro-Winkler	PLSC
Nom normalisé	Exactitude	41,47	55,78	63,40	63,47	63,75
nom sans mots outils	Exactitude	44,15	56,91	64,46	65,23	64,60

Tableau 3.1 – Résultats du couplage du nom des exemples positifs par distance d'édition

On observe le même phénomène pour les autres algorithmes lorsqu'on utilise le nom sans mots outils, soit une amélioration d'environ 1 % à 3 %. Cela peut notamment être dû à l'omission d'un déterminant lors de la saisie de l'information. Par exemple, le risque commercial à comparer « immeubles jeffrey sole realties » est enregistré sous le nom « les immeubles jeffrey sole realties » auprès du Registraire des entreprises.

Cet effet est d'ailleurs notable lorsque l'on compare les algorithmes de Jaro et de Jaro-Winkler, où le second est un dérivé du premier accordant plus d'importance aux premiers caractères des noms. On observe qu'en utilisant le nom normalisé, les résultats sont presque identiques. Par contre, en utilisant la seconde variante du nom, on observe une amélioration de l'exactitude d'environ 1 % avec l'algorithme de Jaro-Winkler. Cette situation est due à la nature de l'utilisation des mots outils. En effet, il est plus probable de retrouver un mot outil au début qu'ailleurs dans le nom, comme le déterminant au début du nom « Les entreprises ». En retirant ces mots en début de nom, on permet à l'algorithme de Jaro-Winkler d'attribuer une plus forte similarité locale sur un mot important en début de nom plutôt que sur un mot outil. Par exemple, le Tableau 3.2 présente la différence entre les similarités de Jaro et de Jaro-Winkler en présence d'un mot outil au début de nom. Puisqu'on obtient de meilleurs résultats à tout coup avec l'algorithme de Jaro-Winkler, nous ne considérons plus l'algorithme de Jaro dans le reste du chapitre.

Noms	Jaro	Jaro-Winkler
les immeubles jeffrey sole realties, les immeubles jeffrey sole inc.	0,78	0,78
immeubles jeffrey sole realties, immeubles jeffrey sole inc.	0,83	0,88

Tableau 3.2 – Exemple de similarité entre deux noms avec ou sans mots outils

3.2.2 Approche à jetons

Le Tableau 3.3 présente les résultats des approches à jetons. On observe que les meilleurs résultats sont obtenus avec les algorithmes de Jaccard et du cosinus, et que ceux-ci sont très similaires peu importe la variante du nom. Les résultats les plus bas sont obtenus par l'algorithme du CSS, qui sont environ 10 % inférieurs aux précédents.

L'amélioration des résultats obtenus en utilisant le nom sans mots outils plutôt que le nom normalisé n'est pas aussi importante avec l'approche à jetons qu'avec la distance d'édition. En effet, on remarque que l'utilisation du nom sans mots outils permet d'améliorer les résultats uniquement avec les algorithmes de Jaccard et du cosinus, et ce de moins de 1 %. Pour les deux autres algorithmes (CSS et MASI), on observe même une diminution entre 1 % et 3 % des résultats.

	(%)	Jaccard	Cosinus	CSS	MASI
Nom normalisé	Exactitude	65,73	65,73	54,09	63,68
nom sans mots outils	Exactitude	66,50	66,36	51,13	62,13

Tableau 3.3 – Résultats du couplage du nom des exemples positifs par jetons

La diminution des résultats la plus notable avec la seconde variante du nom est lorsqu'on utilise le CSS. Le raccourcissement des séquences entraîne ici une baisse des résultats. En effet, la faiblesse du CSS est son dénominateur qui correspond au nombre de jetons de mot dans les noms. Ce dénominateur est de 1 lorsque le nom est composé d'un seul mot, on obtient donc une similarité parfaite dès que ce jeton de mot se retrouve dans le second nom. Par ailleurs, puisque l'algorithme de couplage (algorithme 1) définit le meilleur couplage comme étant la similarité strictement plus grande que la meilleure similarité obtenue jusque-là. Ainsi, la première similarité parfaite trouvée sera automatiquement considérée comme la meilleure. Par exemple, le risque commercial « collins barrow ottawa llp eastern ontario tax service » est positivement couplé avec l'entreprise « collins barrow ottawa llp » avec une similarité de 1 en utilisant le nom normalisé. Toutefois, lorsqu'on utilise la variante du nom sans mots outils, ce risque commercial est associé avec le nom d'entreprise du REQ « service » puisqu'il s'agit de la première entité ayant obtenu une similarité de 1. Cette dernière est de 1 puisqu'elle correspond au ratio de l'intersection du nombre de mots similaires entre les deux chaînes de caractères (c.-à-d. « service ») et le nombre de mots dans la chaîne de caractères qui en contient le moins (c.-à-d. 1), soit $\frac{1}{1} = 1$. Cela signifie qu'un nom sans mots outils composé d'un seul mot couramment utilisé dans le corpus sera automatiquement considéré comme le plus similaire. Par exemple, 88 risques commerciaux ont été couplés avec le nom sans mots outils « inc. » alors que dans l'ensemble des couplages (sans égard à la variante du nom ou de l'algorithme), une entité du REQ n'a été couplée qu'un maximum de trois fois. Pour toutes ces raisons l'algorithme du CSS n'est pas approprié pour notre problématique. Nous ne le considérons donc plus dans le reste du chapitre.

Par ailleurs, on observe qu'avec le MASI, on obtient des résultats d'environ 2 % à 4 % inférieurs à ceux de Jaccard seul. Cela indique que le facteur de pondération M n'améliore pas le couplage des risques commerciaux. Le Tableau 3.4 présente les moyennes des similarités lors d'un couplage avec les algorithmes de Jaccard et du MASI. On y observe que les similarités moyennes obtenues par le MASI sont environ 11 % inférieures à celles de Jaccard. Cela indique

que le facteur M pénalise fortement les comparaisons des noms d'entités. Puisque les résultats du MASI sont inférieurs à ceux obtenus par l'algorithme de Jaccard, nous ne le considérons plus dans le reste du chapitre.

	Jaccard	MASI
Nom normalisé	0,7508	0,6159
nom sans mots outils	0,7633	0,6515

Tableau 3.4 – Moyenne des similarités des couplages de Jaccard et du MASI

3.2.3 Approche avec Monge Elkan

Le Tableau 3.5 présente les résultats des algorithmes de Monge Elkan et de Levenshtein. Le nom sans mots outils est la variante du nom qui permet d'obtenir les meilleurs résultats pour l'algorithme de Monge Elkan. Ces résultats sont cohérents avec la définition de ce dernier, étant donné que celui-ci se base sur Levenshtein. Toutefois, ils sont 2 % inférieurs à ceux rapportés par l'algorithme de Levenshtein en utilisant la même variante du nom.

La piètre performance de Monge Elkan peut s'expliquer par le fait que celui-ci a été conçu pour traiter de longues chaînes de caractères (Cohen *et al.*, 2003a), hors les chaînes traitées ici sont relativement courtes, soit 20 caractères en moyenne. En raison de ces mauvais résultats, nous ne considérons plus cet algorithme dans le reste du chapitre.

	(%)	Levenshtein	Monge Elkan
Nom normalisé	Exactitude	**55,78**	53,60
nom sans mots outils	Exactitude	**56,91**	54,87

Tableau 3.5 – Résultats du couplage du nom des exemples positifs avec les algorithmes de Levenshtein et de Monge Elkan

3.2.4 Sommaire de la section

Les expérimentations démontrent que l'utilisation du nom sans mots outils permet d'obtenir les meilleurs résultats pour les algorithmes retenus jusqu'ici, soit de la CCC, de Levenshtein, de Jaro-Winkler, de la PLSC, de Jaccard et du cosinus. Puisque le nom sans mots outils permet d'obtenir les meilleurs résultats, le nom normalisé n'est plus considéré dans le reste de ce mémoire.

3.3 Couplage d'exemples positifs et négatifs

Les méthodes dont les résultats ont été présentés dans la section précédente utilisent uniquement les entreprises ayant nécessairement une entité présente dans le REQ. Toutefois, comme

nous l'avons constaté lors de l'annotation manuelle des données, certaines entreprises n'ont pas d'entité-vérité dans le REQ. Celles-ci doivent néanmoins être traitées. L'algorithme de couplage doit donc être en mesure d'identifier ces exemples négatifs afin de ne pas les coupler. À cette fin, nous utilisons une fonction de décision pour rejeter un couplage lorsque sa similarité est trop basse.

Pour cette expérience, nous utilisons les 1 706 risques commerciaux de notre banque d'exemples (positifs et négatifs).

3.3.1 Utilisation d'un seuil minimal de similarité

Nous avons ajouté une fonction de décision à l'algorithme 1, soit un seuil minimal de similarité rejetant tout couplage dont la similarité est en deçà du seuil défini. L'algorithme 2 décrit comment effectuer le couplage en utilisant un seuil de similarité.

Algorithme 2 Couplage du nom d'un risque commercial avec fonction de décision

```
1:  procédure MEILLEUR COUPLAGE(risque, seuil_minimale)
2:      meilleure_similarité ← 0
3:      meilleur_couplage ← aucun
4:      entités_candidates ← requête_index(risque.nom)
5:      pour chaque entité_REQ de entités_candidates faire
6:          nom ← entité_REQ.nom
7:          similarité ← calculer_similarité(risque.nom, nom)
8:          si similarité ≥ seuil_minimale et similarité > meilleure_similarité alors
9:              meilleure_similarité ← similarité
10:             meilleur_couplage ← entité_REQ
11:     retourne meilleur_couplage
```

En utilisant cette approche, nous empêchons le couplage d'entités ayant une faible similarité. Toutefois, nos exemples contiennent certains risques commerciaux couplés positivement avec leur entité-vérité, mais dont la similarité est assez faible. Le choix d'un seuil minimal de similarité aura donc un effet sur les résultats, car une valeur trop élevée pourrait certes permettre d'avoir une meilleure précision, mais aurait aussi pour effet de réduire le rappel. À cet fin, une recherche avec différents seuils a été effectuée pour sélectionner celui qui permet d'obtenir les résultats optimaux. Puisque l'objectif est de réduire le nombre de questions de la proposition d'assurance pour le plus de risques commerciaux possible, un résultat optimal est celui dont le rappel est le plus élevé.

Toutefois, la précision et l'exactitude demeurent importantes puisqu'un couplage avec la mauvaise entité ne permet pas de réduire le nombre de questions. Il est donc important d'optimiser ces deux métriques. Dans plusieurs cas, en réduisant d'au plus 1 % le rappel, il est possible d'obtenir une amélioration combinée d'au moins 4 % de l'exactitude et de la précision. Nous avons donc considéré ces résultats comme optimaux. Par exemple, en utilisant le seuil de 0,7

avec Jaro-Winkler, on observe une amélioration de la précision et de l'exactitude de 6,99 % et de 5,68 % respectivement, contre une baisse de 0,28 % du rappel par rapport aux résultats observés avec le précédent seuil maximisant le rappel, soit 0,6.

Le Tableau 3.6 présente les seuils que nous avons retenus après avoir testé l'ensemble des valeurs comprises entre 0,1 et 0,9 par bonds de 0,1. La CCC étant binaire (c.-à-d. qu'elle génère des similarité de 1 ou de 0), aucun seuil n'a été utilisé puisque ce dernier aurait été inutile. De plus, pour Levenshtein, aucune seuil n'a permis d'obtenir de meilleurs résultats sans fortement réduire le rappel.

CCC	Levenshtein	Jaro-Winkler	PLSC	Jaccard	Cosinus
0,0	0,0	0,7	0,4	0,4	0,5

Tableau 3.6 – Seuils minimaux optimaux par algorithme pour le couplage du nom

La Figure 3.1 présente l'ensemble des résultats des algorithmes lorsqu'on utilise ceux-ci pour coupler les 1 706 exemples annotés positivement ou négativement. On y observe deux tendances.

Premièrement, en utilisant un seuil minimal de similarité, nous empêchons le couplage d'un risque commercial avec une entreprise du REQ lorsque leur similarité est en deçà de ce seuil (rouge). Cette approche explique l'amélioration observée pour la précision et l'exactitude. Toutefois, pour obtenir cette amélioration, il est parfois nécessaire de diminuer le rappel puisque certains couplages positifs se situent aussi en deçà du seuil. Puisque les seuils de la PLSC, de Jaccard et du cosinus sont de 0,4 pour les deux premiers, et de 0,5 pour le dernier, et qu'ils ont fait baisser le rappel, nous pouvons conclure que les similarités sont souvent très proches les unes des autres pour un exemple positif ou négatif.

Deuxièmement, en introduisant des exemples négatifs dans notre jeu de données, nous observons pour la majorité des algorithmes une diminution importante de l'exactitude, soit d'environ 11 % sans utiliser de seuil (bleu), et d'environ 9 % en utilisant un seuil (rouge). Puisqu'avec les données positives seules il n'était possible d'avoir que deux types d'erreurs de classification, soit faux négatifs ou couplage avec la mauvaise entité du REQ, l'exactitude était toujours identique au rappel et la précision était de 100 %. En ajoutant les exemples négatifs, nous avons maintenant deux classes à prédire plutôt qu'une. En effet, l'algorithme peut maintenant faire des erreurs de faux positifs et de faux négatifs, c'est-à-dire coupler une entreprise n'ayant pas d'entité-vérité avec une entité du REQ, ou l'inverse. Cela permet d'étudier la précision et le rappel.

De plus, ce sont ces exemples négatifs qui font diminuer l'exactitude de la plupart des algorithmes. L'utilisation d'un seuil permet certes de réduire quelque peu le nombre d'erreurs, mais celui-ci reste inférieur à ceux obtenus précédemment, ce qui démontre que pour plusieurs en-

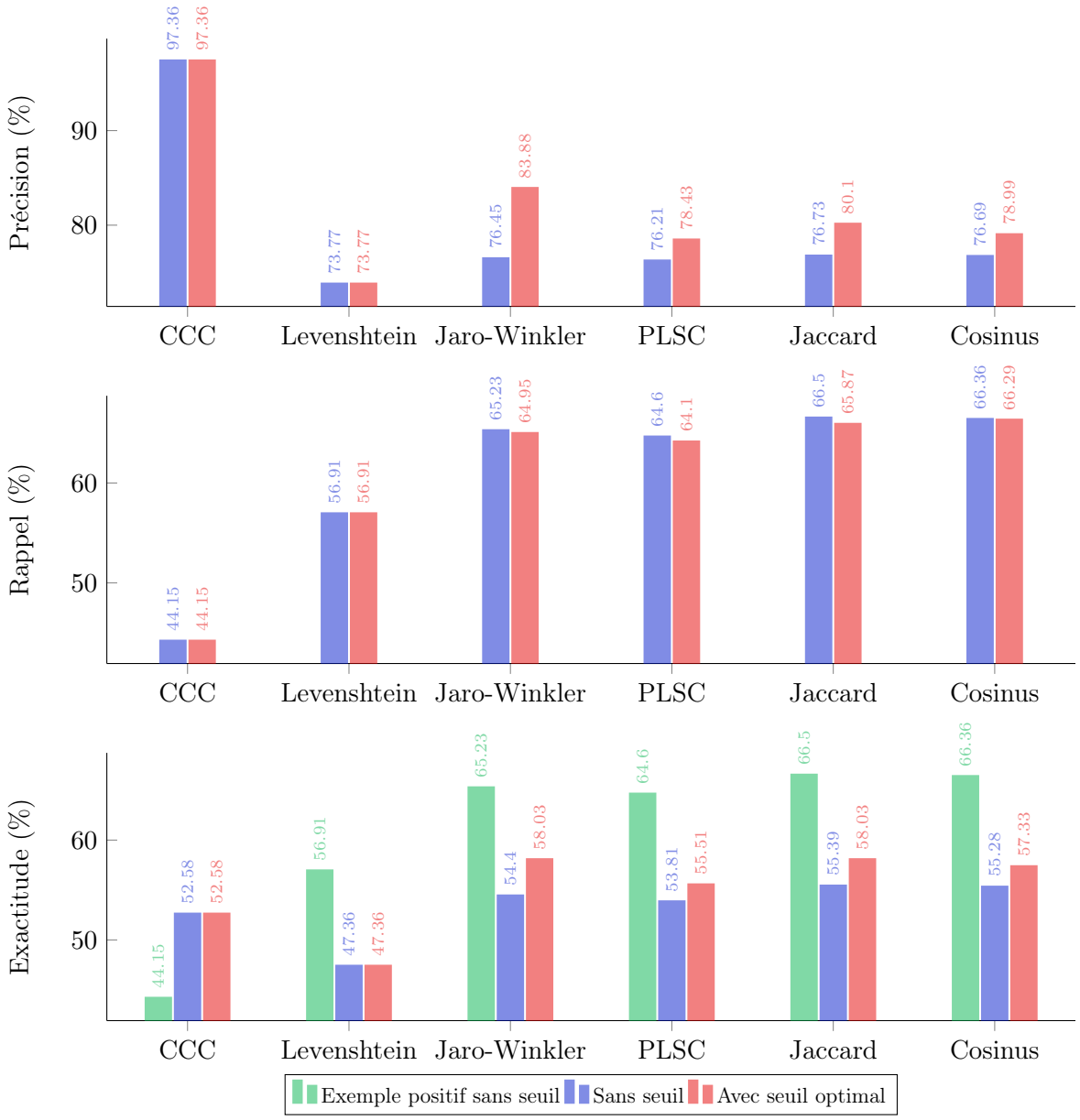

FIGURE 3.1 – Résultats du couplage du nom des exemples positifs et négatifs

treprises annotées positivement ou négativement, il existe plusieurs entreprises du REQ ayant un nom similaire.

Finalement, on observe que la CCC est le seul algorithme de similarité ayant permis une amélioration de l'exactitude lors de l'utilisation de l'ensemble de données d'exemples positifs et négatifs. Cela est dû au fait que l'algorithme de couplage retourne une absence d'entité lorsqu'aucune entité comparée n'a obtenu une similarité au moins plus grande que zéro. En effet, dans le cas de la CCC, la presque totalité (271) des exemples négatifs a été adéquatement négativement annotée ce qui explique la hausse de l'exactitude.

3.4 Discussion sur les résultats obtenus

Nous ne sommes pas en mesure d'obtenir une précision parfaite, même en utilisant la CCC. En effet, environ 15 % des entreprises utilise le prénom et le nom du propriétaire, comme par exemple « David Beauchemin ». Ce type de nom d'entreprise étant commun, il pourrait se retrouver plusieurs fois dans le REQ. Cela signifie qu'un risque commercial peut être couplé par la CCC à une entité du REQ qui porte le même nom, mais qui ne correspond pas à l'entité-vérité. Cette utilisation des prénom et nom du propriétaire d'une entreprise individuelle étant pratique courante, il peut y avoir eu, au fil du temps, plusieurs entreprises enregistrées sous le même nom. Par exemple, il existe cinq entreprises distinctes dont le nom est « David Beauchemin ». Puisqu'il est complexe de quantifier le nombre d'entités du REQ dont le nom correspond à celui de son propriétaire, il est difficile de déterminer l'impact de ces entités sur les résultats du couplage.

3.5 Sommaire du chapitre

Dans ce chapitre, nous avons observé que l'utilisation de la variante du nom sans mots outils comme attribut commun de comparaison entre les risques commerciaux et les entités du REQ permet le couplage adéquat d'au plus 65 % des entreprises. Les résultats indiquent aussi que ce sont les algorithmes de Jaccard et du cosinus qui permettent de coupler correctement le plus de risques commerciaux. Lorsque nous avons utilisé le jeu de données composé uniquement de données positives, nous avons retenu seulement six algorithmes parmi ceux présentés, soit de la CCC, de Levenshtein, de Jaro-Winkler, de la PLSC, de Jaccard et du cosinus.

De plus, nous avons aussi introduit une méthode permettant d'améliorer les résultats, soit l'utilisation d'un seuil minimal de similarité pour chaque algorithme.

Finalement, bien que l'utilisation du nom permette d'obtenir de bons résultats, cette approche n'a pas permis de coupler un nombre suffisant d'entreprises. De plus, nous avons identifié quelques problématiques dans l'utilisation de cet attribut. Mentionnons notamment la présence de plusieurs entreprises différentes portant le même nom. Ces difficultés pourraient être évitées en utilisant l'adresse plutôt que le nom pour associer les entités. Une approche permettant de coupler les entités par l'adresse est abordée dans le prochain chapitre.

Chapitre 4

Couplage de l'adresse par mesure de similarité

Dans ce chapitre, nous présentons les résultats du couplage de l'adresse des risques commerciaux en utilisant les mesures de similarité présentées à la section 2.4. De plus, nous introduisons une méthode permettant d'utiliser les composantes des adresses plutôt que les adresses complètes.

4.1 Couplage de l'adresse

Pour coupler les risques commerciaux à partir de leur adresse, nous utilisons l'algorithme 1 en substituant les noms par l'adresse dans ce dernier. Cela permet d'effectuer le couplage entre une variante de l'adresse d'un risque commercial avec sa plus proche entité du REQ.

Variantes des adresses

Pour effectuer le couplage, nous utilisons soit l'adresse complète normalisée, soit les composantes de l'adresse. La première variante d'une adresse de l'entité correspond à :

— la mise en minuscules,

— la suppression des espaces superflus au début et à la fin de l'adresse,

— le retrait des accents et des caractères spéciaux (p. ex. ç, â, é, ').

Par exemple, lorsqu'on utilise cette procédure de normalisation sur l'adresse « 2325 -1 Rue de l'Université, G1V 0A6 » on obtient « 2325 -1 rue de l'universite, g1v 0a6 ».

La seconde variante de l'adresse implique une étape d'étiquetage des composantes de l'adresse normalisée. Pour effectuer cet étiquetage, l'adresse est segmentée en composantes distinctives par une espace, puis la séquence de composantes obtenue est passée à un marqueur séquentiel qui attribue une étiquette (p. ex. numéro civique) à chacune des composantes. Nous avons

nous-même développé le marqueur séquentiel [1] qui a été utilisé pour effectuer l'étiquetage pour cette tâche étant donné qu'aucune option disponible n'était en mesure de respecter les contraintes de notre problème. Les informations sur ce marqueur sont disponibles à l'annexe B [2]. Les composantes sont ensuite regroupées par types d'étiquette commune. Par exemple, on obtient la variante présentée au Tableau 4.1 lorsqu'on utilise cette procédure de normalisation sur « 2325 -1 rue de l'universite, quebec, qc, g1v 0a6 » [3].

Numéro civique	Numéro d'unité	Nom de rue	Points cardinaux	Ville	Code postal
2325	1	rue de l'universite	∅	quebec	g1v 0a6

Tableau 4.1 – Exemple de segmentation et de regroupement d'une adresse

Pour calculer la similarité entre les composantes de l'adresse, nous utilisons la moyenne pondérée de la similarité entre chacune des composantes individuelles des adresses comparées. Autrement dit, la moyenne de la similarité entre les numéros d'immeubles, les noms de rue, les points cardinaux, les numéros d'unité et les codes postaux. Par exemple, la similarité entre les adresses « 2325 Rue de l'Université G1V 0A6 » et « 2900 Boulevard Edouard-Montpetit H3T 1J4 » correspond à la moyenne de

— sim(2325, 2900),

— sim(rue de universite, boulevard edouard-montpetit),

— $2 \times \text{sim}(\emptyset, \emptyset)$,

— sim(g1v 0a6, h3t 1j4),

où la similarité entre deux ensembles vides ($\emptyset$) correspond à 0.

4.2 Couplage des exemples positifs

L'objectif de cette section est d'analyser les résultats du couplage des 1 418 variantes de l'adresse des exemples positifs en utilisant la comparaison par chaîne de caractères (CCC), de Levenshtein, de Jaro, de Jaro-Winkler, de la plus longue sous-séquence commune (PLSC), de Jaccard, du cosinus, du coefficient de Szymkiewick-Simpson (CSS), du MASI et de Monge Elkan.

4.2.1 Approches par distance d'édition

Le Tableau 4.2 présente les résultats du couplage des adresses en utilisant les approches par distance d'édition. On observe que les meilleurs résultats sont obtenus avec les algorithmes de Jaro et de Jaro-Winkler. Ces derniers permettent d'obtenir une exactitude entre 48 % et

1. Ce marqueur a été développé en collaboration avec Marouane Yassine, un membre du GRAAL. Nous n'avons pas intégré cette portion par respect pour la contribution partagée entre nous deux.

2. Ces travaux sont par ailleurs l'objet d'une publication en cours. L'article, en processus de révision, est présenté à l'annexe C.

3. À noter que nous retirons la province puisque toutes nos données sont au Québec.

52 % selon la variante de l'adresse. Les résultats les plus bas sont obtenus par la CCC avec une exactitude d'environ 13 %.

De plus, on observe qu'en utilisant l'adresse complète normalisée avec la CCC, aucun couplage n'est effectué, ce qui indique qu'aucune adresse des entités-vérités des entreprises n'est écrite exactement de la même façon dans le REQ. Toutefois, environ 13 % de ces entités-vérités ont les mêmes composantes d'adresse. On remarque que l'utilisation de la seconde variante de l'adresse permet aussi d'obtenir de meilleurs résultats pour les autres algorithmes. Cette situation est due à deux raisons : la présence d'abréviation dans les adresses et le positionnement des éléments.

	(%)	CCC	Levenshtein	Jaro	Jaro-Winkler	PLSC
Adresse complète normalisée	Exactitude	0,00	42,31	48,03	48,10	46,33
Composantes de l'adresse	Exactitude	13,19	48,31	51,83	51,83	51,69

Tableau 4.2 – Résultats du couplage de l'adresse des exemples positifs par distance d'édition

Il existe plusieurs abréviations couramment utilisées pour décrire certains éléments de l'adresse. Par exemple, 88,07 % des risques commerciaux utilisent l'abréviation « qc » alors que les entités du REQ utilisent le nom complet « (quebec) ». Cette différence de choix d'écriture de l'information se reflète dans la similarité entre les adresses. Par exemple, la similarité de Jaro-Winkler entre « qc » et « (quebec) » est de 0,54 même s'il s'agit de la même information.

Aussi, le positionnement des éléments de l'adresse n'est pas uniforme entre les deux sources de données. Par exemple, le REQ place le numéro d'unité avant le numéro civique alors que les risques commerciaux font l'inverse. Cette différence de positionnement à un effet sur les algorithmes de distance d'édition puisque le positionnement des caractères est important dans leur cas. Par exemple, la similarité de Jaro-Winkler entre « 1 2425 » et « 2425 1 » est de 0,78.

En utilisant les composantes de l'adresse, on réduit l'impact de la présence d'abréviations et du positionnement des composantes dans la séquence puisque que l'on compare les mêmes éléments entre eux sans égard à leur position dans l'adresse. Par exemple, la similarité de Jaro entre les adresses complètes normalisées « 2325 1 rue de l'universite, g1v 0a6 » et « 1 2325 rue de l'universite, g1v 0a6 » est de 0,85, mais en utilisant les composantes de l'adresse elle est plutôt de 1. Cette approche permet le couplage de près de 13 % plus de risques commerciaux avec l'algorithme de la CCC.

De plus, les résultats quasi identiques observés entre les algorithmes de Jaro et de Jaro-Winkler indiquent que ce qui permet de mesurer la similarité des adresses ne se trouve pas en début de celles-ci, Jaro-Winkler donnant plus de poids aux quatres premiers caractères d'une composante d'adresse. Cela est cohérent avec la structure d'une adresse complète normalisée puisqu'une adresse ne se définit pas uniquement par ses quatre premiers caractères, soit gé-

néralement le numéro civique. Il en va de même pour les autres composantes d'adresse étant donné que l'on en retrouve souvent de très courtes (moins de quatre caractères). En effet, selon le Tableau 4.3, trois composantes sur cinq sont d'une longueur plus courte que la fenêtre de quatre caractères utilisée par Jaro-Winkler. Cela démontre qu'il n'est pas pertinent d'utiliser cet algorithme dans le contexte du couplage de l'adresse, alors il ne sera plus utilisé pour le reste du chapitre.

	Numéro civique	Numéro d'unité	Nom de rue	Points cardinaux	Code postal
Longueur moyenne	3,31	1,06	14,13	0,18	6,00

Tableau 4.3 – Longueur moyenne par composante d'adresse pour l'ensemble des adresses

4.2.2 Approches par jetons

Le Tableau 4.4 présente les résultats du couplage de l'adresse en utilisant les approches par jetons. On observe que ce sont les algorithmes de Jaccard et du CSS qui permettent d'obtenir les meilleurs résultats.

De plus, on observe la même amélioration des résultats décrite précédemment avec les algorithmes de distance d'édition en utilisant les composantes des adresses. Toutefois, cette amélioration est de moitié moins importante que celle de la sous-section 4.2.1 soit environ 3 %.

	(%)	Jaccard	Cosinus	CSS	MASI
Adresse complète normalisée	Exactitude	48,59	48,52	45,91	47,67
Composantes de l'adresse	Exactitude	51,69	51,69	52,19	51,55

Tableau 4.4 – Résultats du couplage de l'adresse des exemples positifs par jetons

Cette amélioration moindre est due à l'importance de la position des mots dans l'adresse. En effet, la position des jetons de mot compte moins pour les algorithmes à jetons puisque ces derniers comparent, pour la plupart, la présence ou non des mêmes jetons dans les chaînes de caractères. Par exemple, la similarité du CSS entre « 1 2425 » et « 2425 1 » correspond à

$$\frac{\{1; 2425\}}{\min(|\{1; 2425\}|, |\{2425; 1\})} = \frac{2}{2} = 1.$$

Finalement, on remarque que les résultats de l'algorithme du MASI sont encore une fois inférieurs à ceux obtenus avec celui de Jaccard seul. Cela signifie que le facteur de pondération M n'améliore pas le couplage des risques commerciaux. Puisque cette approche ne permet pas d'obtenir de meilleurs résultats, elle n'est plus considéré pour le reste de ce mémoire.

4.2.3 Approche par Monge Elkan

Le Tableau 4.5 présente les résultats des algorithmes de Monge Elkan et de Levenshtein. On y observe que l'exactitude lorsqu'on utilise les composantes de l'adresse avec Monge Elkan est environ 1 % supérieure qu'avec Levenshtein seul. Ainsi, puisque l'algorithme hybride de Monge Elkan prend le double du temps pour obtenir des résultats à peine meilleurs que ceux de Levenshtein, nous avons choisi de ne pas le considérer pour le reste de ce mémoire.

	(%)	Levenshtein	Monge Elkan
Adresse complète normalisée	Exactitude	**42,31**	20,38
Composantes de l'adresse	Exactitude	48,31	**49,93**

Tableau 4.5 – Résultats du couplage de l'adresse des exemples positifs avec Levenshtein et Monge Elkan

4.2.4 Sommaire de la section

Les résultats indiquent que l'utilisation des composantes des adresses permet d'obtenir les meilleurs résultats pour les algorithmes retenus, soit la CCC, de Levenshtein, de Jaro, de la PLSC, de Jaccard, du cosinus et du CSS. Puisque cette variante de l'adresse permet d'obtenir les meilleurs résultats, l'adresse normalisée n'est plus considérée dans le reste de ce mémoire.

4.3 Couplage d'exemples positifs et négatifs

L'objectif de cette section est le même que celui de la section 3.3, soit d'analyser les résultats du couplage des exemples positifs et négatifs en utilisant les algorithmes de la CCC, de Levenshtein, de Jaro, de la PLSC, de Jaccard, du cosinus et du CSS.

4.3.1 Utilisation d'un seuil minimal de similarité

Pour effectuer le couplage des exemples positifs et négatifs à l'aide d'une fonction de décision, nous utilisons l'algorithme 2 présenté à la section 3.3. Par contre, nous substituons les noms par l'adresse et utilisons les valeurs de seuils présentées au Tableau 4.6.

Levenshtein	Jaro	PLSC	Jaccard	Cosinus	CSS
0,5	0,3	0,2	0,2	0,2	0,2

Tableau 4.6 – Seuils minimaux optimaux par algorithme pour le couplage de l'adresse

La Figure 3.1 présente l'ensemble des résultats des algorithmes utilisés pour coupler les 1 706 exemples annotés positivement ou négativement. On y observe deux tendances.

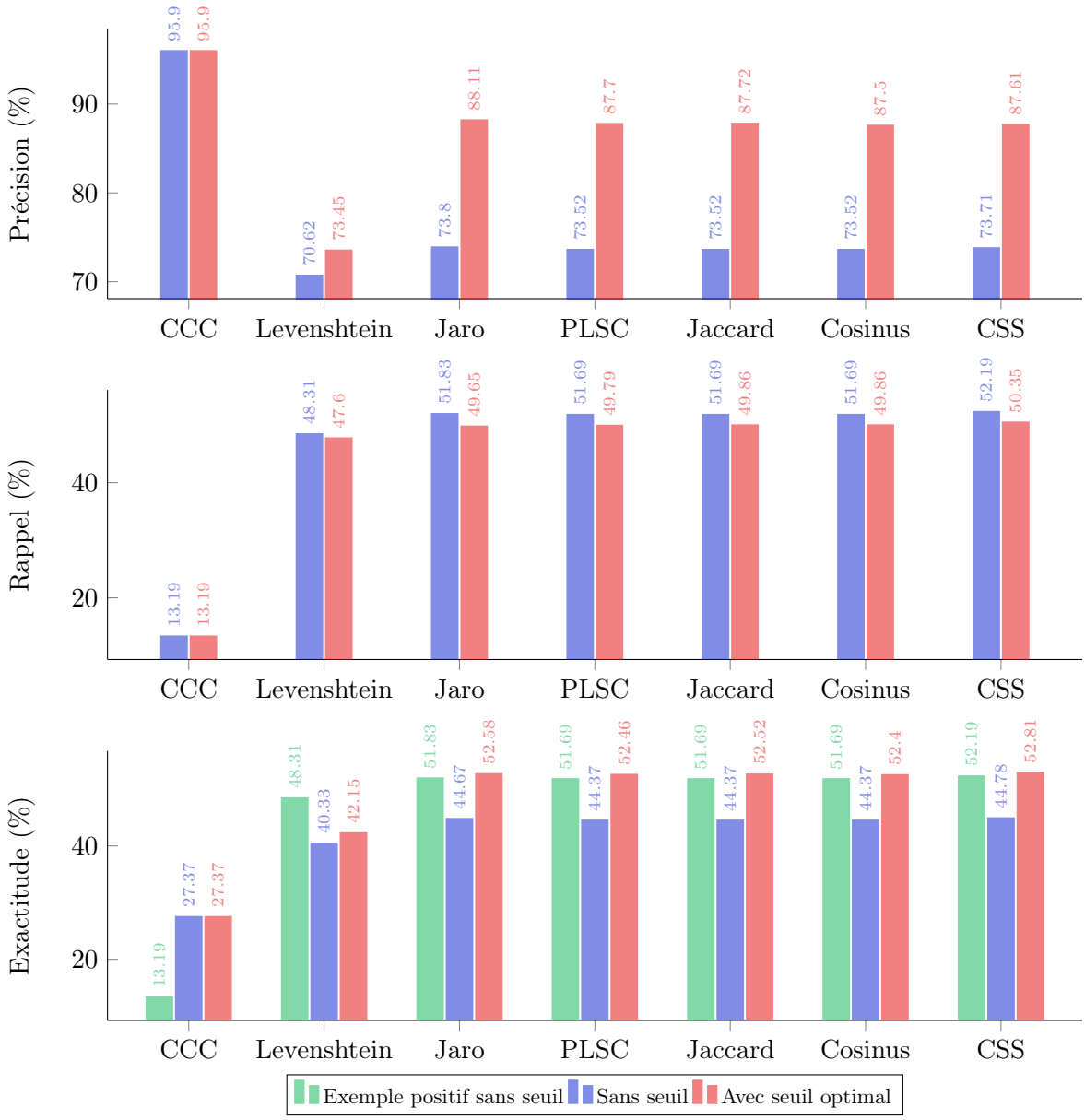

FIGURE 4.1 – Résultats du couplage de l'adresse des exemples positifs et négatifs

Premièrement, en utilisant un seuil minimal de similarité (rouge), on obtient une amélioration de la précision et de l'exactitude d'environ 15 % et 8 % respectivement pour la majorité des algorithmes, mais une diminution d'environ 1 % du rappel. On remarque aussi que les valeurs des seuils optimaux sont relativement basses, mais permettent tout de même une amélioration de la précision et de l'exactitude. Cela indique que pour l'adresse, l'utilisation d'un seuil permet de réduire les mauvais couplages puisque de nombreuses entreprises négativement annotées ont tout de même été couplées correctement malgré une similarité basse.

Deuxièmement, pour la majorité des algorithmes, en utilisant un seuil, on observe une amé-

lioration de l'exactitude d'environ 1 % par rapport aux résultats du couplage des exemples positifs (vert). Cette amélioration de l'exactitude est due au filtrage de quelques entreprises ayant été couplées avec la mauvaise entité, leur similarité étant sous le seuil. L'utilisation de l'adresse et d'un seuil, contrairement au nom seul, permet d'améliorer l'exactitude. Par contre, la qualité des résultats ainsi obtenus demeure inférieure d'environ 5 % par rapport à ceux obtenus avec le nom.

4.4 Discussion sur les résultats obtenus

Dans l'ensemble, les résultats du couplage de l'adresse en utilisant les mesures de similarité sont concluants. Ils indiquent qu'avec les composantes de l'adresse, il est possible de coupler près de 50 % des risques commerciaux.

Ces résultats sont d'autant plus satisfaisants si l'on tient compte du fait qu'environ 36 % des entités du REQ n'ont aucune adresse et que 15,82 % des entités-vérité des risques commerciaux n'en ont pas non plus[4]. Autrement dit, 270 risques commerciaux sont impossible à coupler positivement par l'adresse. Puisqu'il est impossible de coupler positivement ces risques, les résultats de l'exactitude et du rappel ne sont pas représentatifs de l'efficacité des algorithmes. En effet, ces métriques ne peuvent atteindre qu'une valeur d'au plus 77,5 %. Par exemple, on peut observer cet effet sur l'algorithme du CSS dans le Tableau 4.7, où le rappel et l'exactitude augmentent de 11,85 % et 9,93 % respectivement en retirant les risques commerciaux ayant une entité-vérité sans adresse.

(%)	Risques commerciaux avec entité-vérité **ayant une adresse ou non** (1 706 risques)	Risques commerciaux avec entité-vérité **ayant une adresse** (1 436 risques)
Précision	87,61	87,61
Rappel	50,35	62,20
Exactitude	52,81	62,74

Tableau 4.7 – Résultats du couplage des composantes de l'adresse avec le CSS des risques commerciaux avec ou sans entité-vérité ayant une adresse

De plus, on constate que les résultats de la majorité des algorithmes sont très similaires, la différence étant souvent d'environ 1 % entre ces derniers. En plus, les seuils optimaux sont souvent identiques pour les différentes approches, tel que démontré dans le Tableau 4.6. Ce phénomène est attribuable aux calculs des similarités des composantes vides. En effet, lors du calcul de la similarité entre deux adresses par leurs composantes, si l'une de ces dernières est vide, la similarité pour cette composante est égale à zéro. En considérant que 77,85 % et 88,19 % des adresses n'ont pas de numéro d'unité ou de points cardinaux respectivement

4. En raison de la politique de confidentialité du REQ. Lorsque l'adresse d'une entreprise correspond à celle d'un administrateur, elle n'est pas rendue publique (Registraire des entreprises du Québec, 2017).

(Tableau 4.8), on peut déduire que plus de la majorité des similarités entre deux adresses ne peuvent avoir une valeur plus élevée que 0,8.

(%)	Numéro civique	Numéro d'unité	Nom de rue	Points cardinaux	Code postal
Composantes vides	1,13	77,85	0,39	88,19	0,05

Tableau 4.8 – Pourcentage de composantes d'adresses vides pour l'ensemble des adresses

Par exemple, seulement deux couplages positifs ont obtenu une similarité supérieure à 0,8 avec les algorithmes à jetons. Pourtant, 73 adresses ont été couplées positivement par l'algorithme de la CCC, ce qui indique qu'au moins 71 autres adresses auraient dû obtenir une similarité au moins supérieure à 0,8. Cette situation est attribuable à la méthode utilisée pour calculer la similarité des composantes d'adresse. En effet, puisqu'on utilise la moyenne pondérée des similarités avec des poids uniformes de 0,2 par composante, lorsque l'une d'elles est vide, la moyenne diminue automatiquement de 0,2 par composante vide.

4.5 Sommaire du chapitre

Dans ce chapitre, les résultats obtenus avec l'adresse normalisée étaient minés par l'absence d'uniformité dans la représentation de l'adresse entre les sources de données. Toutefois, en utilisant la variante par composante d'adresse, nous avons obtenu de meilleurs résultats. C'est l'algorithme du CSS qui a permis de coupler le plus de risques commerciaux, soit 50 %. Enfin, les résultats permettent d'affirmer que les algorithmes du MASI et de Monge Elkan ne sont pas de bons candidats pour résoudre notre problématique.

Finalement, divers problèmes dans l'utilisation de l'adresse comme unique attribut de couplage ont été relevés dans ce chapitre, notamment l'absence d'adresse pour plusieurs entités du REQ. Puisque cette information n'est pas toujours disponible, il est plus pertinent d'effectuer le couplage par le nom et d'ajouter l'adresse, si possible, afin d'améliorer la capacité de couplage des algorithmes. Pour cette raison, une solution permettant d'utiliser ces deux attributs conjointement est présentée dans le prochain chapitre.

Chapitre 5

Couplage du nom et de l'adresse par apprentissage automatique

Dans ce chapitre, nous présentons les résultats du couplage du nom et de l'adresse des risques commerciaux en utilisant les algorithmes d'apprentissage automatique présentés à la section 2.5, soit la régression logistique, la forêt aléatoire et le perceptron multicouche.

5.1 Couplage du nom et de l'adresse

L'algorithme 3 décrit comment effectuer le couplage du nom et de l'adresse d'un risque commercial. Cet algorithme repose sur le calcul de la similarité entre les attributs sélectionnés à l'aide d'un algorithme d'apprentissage automatique qui utilise un vecteur d'informations généré à partir d'un risque commercial et d'une entité du REQ. L'entité qui obtient la plus haute similarité au-dessus d'un seuil minimal est considérée comme la plus similaire au risque commercial.

Algorithme 3 Couplage du nom et de l'adresse d'un risque commercial

```
1:  procédure MEILLEUR COUPLAGE(risque, seuil_minimale)
2:      meilleure_similarité ← 0
3:      meilleur_couplage ← Aucun
4:      entités_candidates ← requête_index(risque.nom)
5:      pour chaque entité_REQ de entités_candidates faire
6:          noms ← entité_REQ.noms
7:          adresses ← entité_REQ.adresses
8:          similarité ← calculer similarité(risque.nom, risque.adresse, noms, adresses)
9:          si similarité > seuil_minimale et similarité > meilleure_similarité alors
10:             meilleure_similarité ← similarité
11:             meilleur_couplage ← entité_REQ
12:     retourne meilleur_couplage
```

Vecteur d'informations

Pour effectuer le couplage à partir des noms et des adresses, nous utilisons des vecteur d'informations générés à partir de chacune des paires d'entreprises (Martins, 2011). Chaque vecteur sera donc constitué de deux composantes de similarités : le nom et l'adresse.

La première composante est constituée de la combinaison qui donne la similarité maximale, sans seuil, entre le nom sans mots outils du risque commercial et ceux de l'entité du REQ, selon chaque algorithme de similarité retenu au chapitre 3. Par exemple, la composante du nom du vecteur d'informations entre les entreprises A et B (Tableau 5.1) correspond au vecteur présenté dans le Tableau 5.2. Dans cet exemple, les similarités présentées pour chaque algorithme correspondent à la plus grande similarité entre « (universite laval, universite montreal) » et « (universite laval, udem) ».

universite laval	universite de montreal, udem
2325 rue de l'universite g1v 0a6	2900 boulevard edouard-montpetit h3t 1j4
	2900 blvd edouard-montpetit h3t 1j4

Tableau 5.1 – Informations sur l'entreprise A (gauche) et l'entité B (droite)

CCC	Levenshtein	Jaro-Winkler	PLSC	Jaccard	Cosinus
0,0000	0,9474	0,6667	0,5263	0,2500	0,4082

Tableau 5.2 – Similarités maximales du nom entre les entreprises A et B

La seconde composante est constituée de la similarité maximale, sans seuil, entre les composantes de l'adresse du risque commercial et celles de l'entité du REQ, selon chaque algorithme de similarité retenu au chapitre 4. Ainsi, en utilisant les données de l'exemple précédent (Tableau 5.1), la composante de l'adresse correspond ici au vecteur d'informations présenté dans le Tableau 5.3. Ces similarités correspondent donc à la plus grande similarité entre « (2325 rue de universite g1v 0a6, 2900 boulevard edouard-montpetit h3t 1j4) » et « (2325 rue de universite g1v 0a6, 2900 blvd edouard-montpetit h3t 1j4) » par composante de l'adresse.

CCC	Levenshtein	Jaro	PLSC	Jaccard	Cosinus	CSS	MASI
0,0000	0,4710	0,4000	0,4000	0,4000	0,4000	0,0000	0,0000

Tableau 5.3 – Similarités maximales de l'adresse entre l'entreprise A et B

En somme, le vecteur résultant des deux composantes est donc composé de six similarités provenant du nom et huit de l'adresse, pour un total de quatorze attributs. Par la suite, les algorithmes d'apprentissage automatique vont sélectionner les plus utiles.

Les vecteurs sont générés uniquement entre les risques commerciaux et les entités ayant été sélectionnées à partir de l'index des noms uniquement, puisque ce dernier permet d'obtenir la plus haute valeur de rappel possible.

5.2 Entraînement des modèles d'appprentissage automatique

Pour obtenir une similarité à partir d'un vecteur d'informations, nous avons utilisé des modèles d'apprentissage automatique entraînés sur les données annotées. Ces modèles ont été entraînés selon les algorithmes de la régression logistique, de la forêt aléatoire et du perceptron multicouche. Ces algorithmes de classification tentent de prédire la classe à laquelle appartient la paire d'entités, soit la classe 1 s'ils sont la même entité (similarité), et 0 autrement (dissimilarité). À partir de cette tâche de classification, on obtient un score, soit l'appartenance à l'une ou l'autre des deux classes. Étant donné que nous nous intéressons à la similarité entre deux entités, nous utilisons uniquement le score d'appartenance à la classe 1, et nous rejetons l'autre.

La procédure d'entraînement comporte deux étapes : une de recherche en grille des paramètres optimaux par modèle en validation croisée, et une autre de validation de l'intégration du modèle dans l'algorithme de couplage. Pour permettre la reproduction des résultats, nous utilisons la même amorce à chaque étape, soit 42.

Pour effectuer cet entraînement, il faut d'abord séparer les données en deux ensembles : d'entraînement et de test. Nous avons d'abord séparé les 1 706 données annotées selon une approche 80-20. Le plus petit ensemble, qui contient 291 exemples positifs et 51 négatifs, est utilisé comme ensemble de test.

Les 1 364 données restantes seront utilisées comme ensemble d'entraînement. De ces dernières, 1 123 sont positives et 241 sont négatives. Puisqu'il est nécessaire de comparer deux entités pour construire le vecteur d'informations, il faut associer les exemples négatifs avec une entité quelconque. Nous avons donc sélectionné aléatoirement des entreprises du REQ n'ayant pas les mêmes noms et adresses selon la comparaison par chaîne de caractères (CCC) pour être couplées avec les exemples négativement annotés.

Prati *et al.* (2009) ont démontré que l'optimisation des algorithmes d'apprentissage automatique était problématique lorsque les classes ne sont pas équilibrées. Comme nous avons 18 % d'exemples négatifs dans l'ensemble d'entraînement, il faut soit rééquilibrer les données ou augmenter le poids des exemples ayant une plus faible proportion. Ayant à notre disposition un très grand nombre d'entreprises dans le REQ, nous pouvions facilement créer des exemples négatifs en y pigeant au hasard des entités comportant au moins un nom et une adresse. Nous avons sélectionné les premiers nom et adresse d'une entité du REQ pour constituer un risque commercial factice. Celui-ci a ensuite été couplé aléatoirement avec une autre entité du REQ

n'ayant pas le même nom en CCC. Avec cette procédure, nous avons ajouté 882 exemples négatifs pour obtenir un ensemble finale de 2 246 données. Cette méthode n'a pas été appliquée à notre ensemble de test puisque celui-ci n'est pas utilisé pour l'entraînement des algorithmes de classification, mais plutôt pour tester celui de couplage.

Finalement, pour la totalité de la procédure d'entraînement, nous avons appliqué une transformation centrée réduite sur toutes les données. La moyenne et la variance ont été calculées à partir de l'ensemble d'entraînement uniquement.

Recherche en grille

Cette étape consiste à effectuer une optimisation des modèles selon différentes combinaisons de paramètres afin d'obtenir les configurations optimales. Chaque modèle possède son espace de paramètres sur lequel est exécutée une recherche en grille par validation croisée à cinq plis (Refaeilzadeh *et al.*, 2009). En utilisant cette approche de validation statistique, nous pouvons évaluer la fiabilité d'un modèle grâce à une méthode d'échantillonnage qui consiste à diviser l'échantillon original d'entraînement (2 246) en cinq sous-échantillons distincts. Parmi de ces derniers, nous en sélectionnons un pour la validation et les autres pour l'entraînement. Nous répétons ensuite cette étape en permutant les échantillons jusqu'à ce que chacun des plis ait occupé la position d'échantillon de validation. Cela signifie qu'avec cette approche, nous obtenons cinq résultats différents permettant de calculer la moyenne et la variance sur les métriques pour chaque combinaison de paramètres. Ainsi nous pouvons sélectionner la configuration ayant obtenu la meilleure moyenne sur cinq entraînements plutôt que sur un seul.

Les paramètres de la recherche en grille pour la régression logistique sont : le paramètre C, qui correspond à l'inverse de la force de régularisation, et la tolérance à l'arrêt de l'entraînement. Pour le premier paramètre, nous cherchons la meilleure valeur parmi les 1 000 de l'espace logarithmique compris entre -14 et 0. Pour le second, nous cherchons la meilleure valeur parmi les 150 de l'espace logarithmique compris entre -16 et -6. Dans tous les cas, nous utilisons le résolveur LIBLINEAR (Fan *et al.*, 2008) pour un maximum de 100 000 itérations. Les paramètres optimaux obtenus par cette recherche sont de 0,498355789 et $1e{-}16$ respectivement.

Le paramètre de la recherche en grille pour la forêt aléatoire est le nombre d'arbres dans la forêt. Pour ce paramètre, nous cherchons la valeur optimale parmi l'ensemble $\{1024, 2048, 4096, 8192\}$. Le paramètre optimal obtenu par cette recherche est 2048.

Contrairement aux autres recherches en grille, celle du perceptron multicouche se fait de manière aléatoire selon une distribution de paramètres. Nous définissons un espace de distribution dans lequel nous pigeons 1 000 configurations qui sont ensuite utilisées pour l'entraînement. Nous utilisons cette approche puisque notre espace des paramètres est trop vaste. Les paramètres de recherche sont : le nombre de couches, le nombre de neurones par couche et la

tolérance de l'amélioration de la perte.

Pour le premier paramètre, nous cherchons le nombre de couches optimal allant de 3 à 5. Pour le second, nous cherchons le nombre optimal de neurones par couche parmi l'ensemble des nombres allant de 5 à 120 par bonds de 5. Par exemple, lorsque le nombre de couches est de 2, nous obtenons les combinaisons $(5, 5), (5, 10), \ldots, (120, 115), (120, 120)$. Pour le dernier paramètre, nous cherchons à optimiser la tolérance à l'amélioration de la perte lors de l'optimisation parmi les 100 valeurs de l'espace logarithmique compris entre -14 et -1. Dans tous les cas, nous utilisons l'algorithme d'optimisation Adam (Kingma et Ba, 2014) pour un maximum de 10 000 itérations avec une patience sans amélioration de la fonction de perte d'une durée de 25 itérations. De plus, la fonction d'activation des neurones est la fonction logistique. Après la recherche en grille, les paramètres optimaux sont 4 couches avec 20, 95, 20 et 110 neurones respectivement, et la tolérance est de 1,26185e−12.

Validation du modèle de couplage

La dernière étape correspond à la validation de l'algorithme de couplage en utilisant les modèles de classification entraînés avec une approche de validation croisée selon les paramètres optimaux. Cette évaluation utilise l'ensemble de données de validation qui contient 342 risques commerciaux annotés manuellement.

5.3 Résultats et analyse

L'objectif de cette section est d'analyser les différents résultats du couplage du nom et de l'adresse en utilisant les algorithmes d'apprentissage automatique. Nous y présentons les résultats du couplage des exemples positifs et négatifs en utilisant le seuil optimal de similarité ayant été préalablement déterminé par une recherche exhaustive similaire à celle utilisée dans les deux chapitres précédents. Toutefois, nous faisons des bonds de 0,001 plutôt que de 0,1. Les seuils utilisés sont 0,888 pour la régression logistique, 0,96 pour la forêt aléatoire et 0,996 pour le perceptron multicouche.

Les résultats ainsi obtenus sont comparés à ceux de référence, soit la meilleure configuration obtenue jusqu'à présent. Il s'agit de l'algorithme de Jaccard sur la variante du nom sans mots outils avec un seuil de similarité de 0,4 appliqué sur l'ensemble de test.

Le Tableau 5.4 présente les résultats du couplage du nom et de l'adresse des algorithmes d'apprentissage automatique et le résultat du couplage de l'algorithme de référence. On y observe que la forêt aléatoire et le perceptron multicouche obtiennent de meilleurs résultats de rappel que Jaccard. Ces derniers permettent de coupler entre 1 % et 7 % plus d'entreprises. De plus, la précision est moins de 1 % inférieure à celle observée avec l'algorithme de Jaccard, et l'exactitude est au moins aussi bonne. Cela démontre que dans l'ensemble, ces deux algorithmes

de couplage du nom et de l'adresse permettent d'obtenir de meilleurs résultats que lorsqu'on effectue ce couplage sur le nom seulement avec Jaccard.

(%)	Régression logistique	Forêt aléatoire	Perceptron multicouche	Jaccard
Précision	89,77	81,06	87,55	81,78
Rappel	66,67	73,54	79,73	72,51
Exactitude	64,91	62,87	73,10	62,87

Tableau 5.4 – Résultats du couplage du nom et de l'adresse par apprentissage automatique

Par contre, le rappel obtenu avec la régression logistique est 6 % inférieur à celui obtenu avec l'algorithme de Jaccard, et ce malgré le fait que ce dernier algorithme est utilisé dans la construction du vecteur d'informations utilisé par le premier. Toutefois, l'un des avantages de la régression logistique est la possibilité d'interpréter ses coefficients logarithmiques. Ceux-ci sont présentés en format logarithmique dans le Tableau 5.5.

Un coefficient négatif indique que l'attribut a un faible effet sur le classificateur alors qu'un coefficient positif indique le contraire. Notamment, lorsque l'attribut CCC est à 1, le score s'approche de 1. On observe aussi que les poids des variables de l'adresse (au bas du tableau) sont beaucoup plus faibles que ceux du nom (en haut). En effet, le nom occupe un rôle principal pour déterminer si deux entités sont correspondantes et l'adresse permet d'améliorer ces résultats. Ces coefficients démontrent que le modèle est cohérent avec les résultats observés dans les deux chapitres précédents, toutefois il n'a pas bien généralisé sur de nouvelles données, autrement dit, il a fait du sur-apprentissage.

CCC	Levenshtein	Jaro-Winkler	PLSC	Jaccard	Cosinus	CSS
9,87197	5,26087	1,94404	-1,49680	2,94360	1,69059	-
CCC	Levenshtein	Jaro	PLSC	Jaccard	Cosinus	CSS
0,71328	-0,93295	-0,43295	1,82524	0,42491	-0,51590	-0,17981

Tableau 5.5 – Coefficient logistique par attribut de la régression logistique avec ceux du nom en haut et ceux de l'adresse en bas

5.4 Discussion sur les résultats obtenus

L'utilisation des modèles d'apprentissage automatique comme algorithme de classification dans le processus de couplage d'un risque commercial et d'une entité du REQ donne de meilleurs résultats que ceux observés pour la meilleure configuration jusqu'à présent. Les résultats démontrent que l'intuition d'utiliser le nom et l'adresse conjointement afin de coupler les risques commerciaux s'avère pertinente.

Malgré les bons résultats observés, certaines erreurs de couplage sont difficiles à justifier. Par exemple, le Tableau 5.6 présente l'une de ces erreurs obtenue uniquement lors du couplage par régression logistique. On y observe que les noms couplés ne sont pas les mêmes. Cependant, le fait que ces derniers partagent les mots « construction » et « inc. » a pu induire le modèle en erreur. L'adresse en revanche, elle est loin de correspondre. L'une est dans la région de Québec, alors que l'autre est dans la région de Sherbrooke. Pourtant, la régression logistique a attribué une similarité de 0,99934 à ce couplage même si l'entité-vérité est présentée sous le nom très similaire de « Les constructions Alain Cloutier inc. », mais avec une adresse différente, le « 549 2e rang, Sherbrooke J1C 0B1 ».

Risque commercial	Entité
construction alain cloutier inc. 1030 rue de l'ardoise sherbrooke j1c 0j6	construction steeve arbourd inc. 2-1822 rue notre-dame, l'ancienne-lorette, g2e 3c7

Tableau 5.6 – Exemple d'erreur de couplage de la régression logistique

De plus, les valeurs des seuils minimaux de similarité sont très élevées, la plus haute étant celle du perceptron multicouche avec 0,996. Cette observation démontre que les similarités sont très élevées et qu'elles ne sont pas uniformément distribuées entre 0 et 1, mais plutôt concentrées près de 1.

Finalement, malgré les bons résultats observés en utilisant les algorithmes d'apprentissage automatique, l'optimisation de ceux-ci demande beaucoup de finesse puisqu'une mauvaise configuration peut rapidement affecter la qualité des résultats. Par exemple, l'utilisation de la fonction d'activation hyperbolique diminue la qualité des résultats du perceptron multicouche de près de moitié. Il est donc important d'effectuer une exploration rigoureuse des différents paramètres des algorithmes pour trouver la configuration permettant d'obtenir les meilleurs résultats.

5.5 Amélioration des résultats

Les résultats observés à la section 5.3 nous ont permis de démontrer empiriquement que le couplage du nom et de l'adresse à l'aide d'algorithmes d'apprentissage automatique permet d'obtenir de meilleurs résultats que le couplage avec l'algorithme de Jaccard sur le même ensemble de test. Toutefois, cette amélioration demeure peu élevée puisqu'à peine 80 % des risques commerciaux ont été couplés. Diverses solutions sont possibles afin d'améliorer les résultats, notamment l'augmentation de la taille des ensembles de données, l'utilisation d'autres algorithmes d'apprentissage automatique ou de nouveaux attributs communs, et la création de variables explicatives.

Les deux premières solutions ont déjà été appliquées durant les travaux. Nous avons en effet

annoté des nouvelles données à plusieurs reprises et nous avons testé plusieurs autres algorithmes (p. ex. machine à vecteurs de support).

La troisième approche est complexe à mettre en place. Comme nous nous sommes imposé la contrainte de nous baser seulement sur le nom et l'adresse, il nous faut prédire le prochain attribut commun. L'un de ces attributs potentiels est le secteur d'activité économique, puisque celui-ci est disponible dans le REQ et qu'il est possible de le déduire à partir des informations du risque commercial. En effet, Baillargeon (2018) a proposé une méthode permettant de déduire le type d'activité économique d'un risque commercial à partir du nom et de l'adresse. Toutefois, malgré la possibilité et la disponibilité de cet attribut, son utilisation reste difficile à utiliser. En effet, la représentation des secteurs d'activité de l'assureur et du REQ sont divergentes. Premièrement, ils ne sont pas dans la même langue, l'un étant en anglais et l'autre en français. Deuxièmement, la segmentation des secteurs d'activités n'est pas correspondante. Le Registraire des entreprises du Québec divise les entreprises selon des secteurs d'activités couramment utilisés dans des analyses sectorielles (p. ex. industries du textile), alors que l'assureur les divise par objectifs de calcul de la prime d'assurance. Pour pallier cette difficulté, nous avons développé une table d'équivalences. Toutefois, nos expérimentations préliminaires indiquent que cette approche ne permet pas d'améliorer les résultats.

La quatrième et dernière approche est la génération de variables explicatives. Nous avons ici créé 20 nouvelles variables explicatives telles que le nombre de composantes de l'adresse communes en CCC entre les entités, le minimum et le maximum de la différence de longueur entre les noms des entreprises, et le nombre de similarités supérieures au seuil optimal entre les deux entreprises obtenu par chaque algorithme de similarité. Par contre, ces variables se sont souvent avérées peu denses, ce qui a semblé nuire aux résultats. Aussi avons-nous tenté d'utiliser une analyse en composantes principales sur ces nouvelles variables afin d'en augmenter la densité et d'en obtenir une meilleure représentation, mais sans succès.

Toutefois, une cinquième approche est possible, soit l'assouplissement de notre restriction de départ voulant que l'on couple les risques commerciaux avec l'entité la plus similaire uniquement, pour plutôt le faire avec les N plus similaires. Ainsi, si l'entité-vérité se retrouve dans les N similarités les plus élevées, nous considérons qu'il s'agit d'un bon couplage. L'intuition de cette approche est que les similarités générées par les algorithmes sont souvent très élevées et très proches les unes des autres, la similarité moyenne pour chacun des trois algorithmes étant dans l'ordre 0,84, 0,99 et 0,92. Le fait de rejeter toutes ces « meilleures réponses potentielles » pourrait nous faire passer à côté de « la » meilleure.

De plus, puisque l'objectif est de simplifier le remplissage d'un questionnaire client, il est possible d'utiliser une méthode de validation humaine de la bonne réponse parmi les N entités proposées par l'algorithme de couplage. Cette approche permet par ailleurs de faire de l'entraînement par amorçage (*bootstrapping*), qui consiste à générer de nouvelles données qui sont

ensuite utilisées pour effectuer un nouvel entraînement du modèle.

5.5.1 N plus probables

La Figure 5.1 présente les résultats en utilisant des valeurs de $N = 1$, 5 et 10. Elle démontre que cette intuition de proximité entre les similarités pouvant contenir la bonne réponse est valide puisqu'on y observe une amélioration entre 8 % et 13 % du rappel pour l'ensemble des algorithmes, incluant celui de Jaccard.

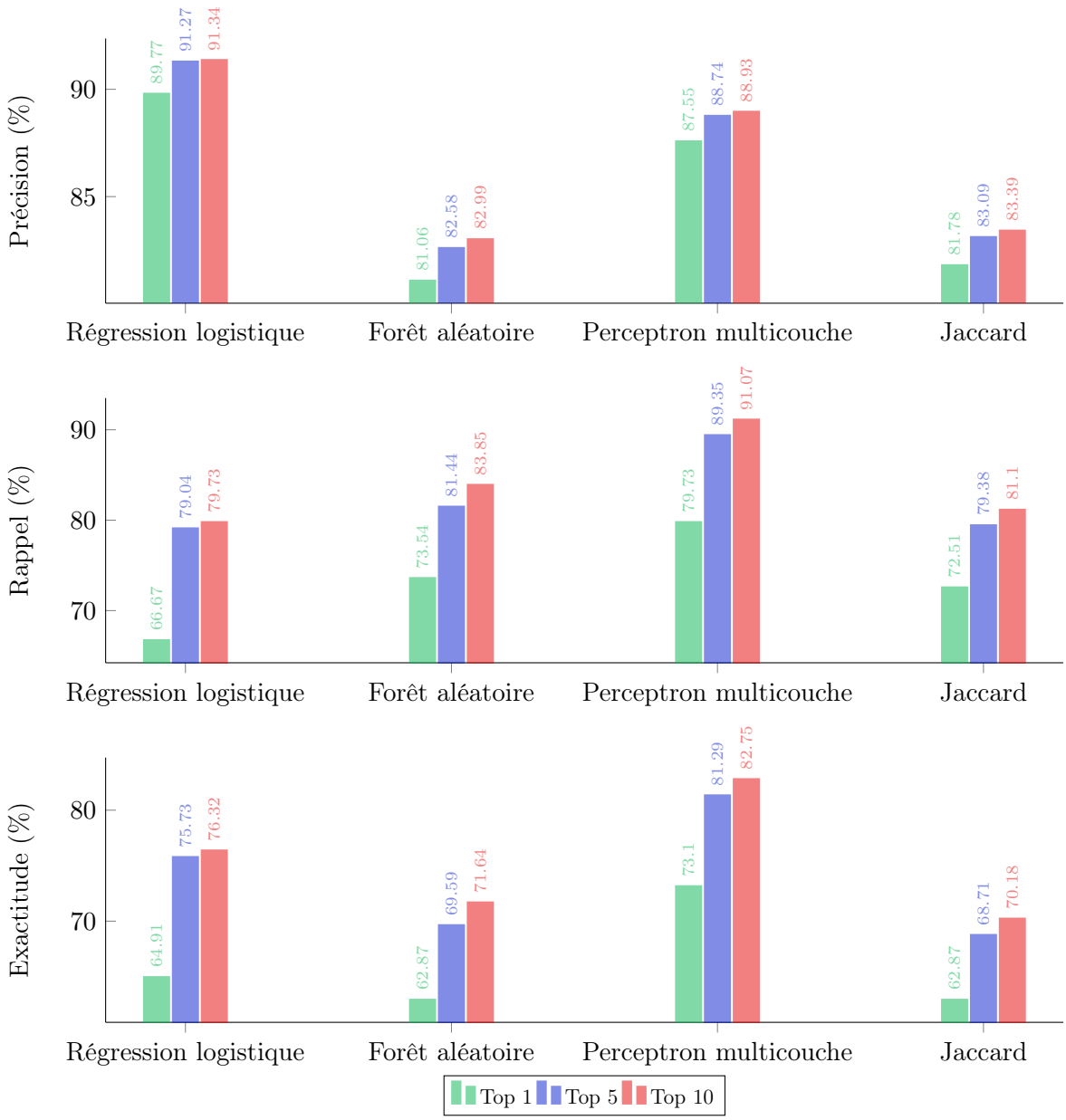

FIGURE 5.1 – Résultats du couplage du nom et de l'adresse par apprentissage automatique des N plus probables

De plus, en utilisant la valeur 10, le perceptron multicouche atteint presque la plus haute valeur de rappel possible (93 %). Rappelons que la performance du rappel est limitée en raison de la recherche à l'aide de l'index. C'est l'utilisation du perceptron multicouche qui permet d'obtenir les meilleurs résultats, avec une précision et une exactitude de près de 89 % et de 83 % respectivement.

Le Tableau 5.7 présente les temps d'inférence de l'ensemble de test des différents modèles en excluant la recherche dans l'index pour l'ensemble des algorithmes. Les chiffres sur les modèles d'apprentissage automatique incluent aussi le temps nécessaire pour la génération des vecteurs de similarité, qui est en moyenne d'environ 1,3 seconde par entreprise. On observe que le modèle prenant le plus de temps est la forêt aléatoire. Cette approche est plus lente puisque nous utilisons 2 048 arbres. Par ailleurs, on observe un temps d'inférence très similaire entre la régression logistique et le perceptron multicouche. Cela démontre que l'utilisation du perceptron multicouche, bien qu'il soit moins interprétable que la régression logistique, permet toutefois d'obtenir de meilleurs résultats dans l'ensemble, et ce dans un temps équivalent. Par contre, la construction du vecteur d'informations requiert l'exécution d'algorithmes de similarité, ce qui nécessite plusieurs minutes de calcul, et implique donc un délai accru par rapport à l'utilisation de l'algorithme de Jaccard. Cependant, la différence de temps de calcul par entreprise avec le perceptron multicouche, l'algorithme ayant obtenu les meilleurs résultats, est d'environ 1 seconde.

(secondes)	Régression logistique	Forêt aléatoire	Perceptron multicouche	Jaccard
Temps	1,32	1,74	1,34	0,25

Tableau 5.7 – Moyenne par entreprise des temps d'inférence des algorithmes d'apprentissage automatique

En somme, en utilisant le perceptron multicouche, nous sommes en mesure de coupler presque toutes les entreprises pouvant l'être suite à une recherche dans un index. Ces résultats sont environ 11 % supérieurs à ceux observés avec la meilleure configuration pour un seul attribut, soit Jaccard avec le seuil optimal de similarité de 0,4, tout en obtenant une inférence des résultats s'approchant de celle obtenue par l'approche la plus simple, soit la régression logistique.

5.6 Sommaire du chapitre

Dans ce chapitre, nous avons démontré empiriquement que le couplage par un algorithme d'apprentissage automatique en utilisant le nom et l'adresse conjointement donne en majorité de meilleurs résultats que le couplage en utilisant un seul attribut à la fois. Toutefois, la sensibilité des algorithmes d'apprentissage à certains paramètres rend leur entraînement plus

complexe. Malgré cela, c'est le perceptron multicouche qui a donné les meilleurs résultats même s'il agit de celui qui a été le plus sensible à l'ajustement de ses paramètres d'entraînement. Dans une tentative finale d'améliorer davantage nos résultats, nous avons utilisé une approche par les N plus probables, ce qui nous a permis d'atteindre presque la valeur maximale de rappel avec le perceptron multicouche.

Dans le prochain chapitre, nous discuterons des avantages et des limites des travaux présentés dans ce mémoire ainsi que des pistes d'exploration possibles.

Conclusion

Dans ce mémoire, nous avons abordé la problématique du couplage des doublons entre deux sources de données afin d'acquérir des informations sur des entreprises lors du processus d'appréciation d'un risque par une compagnie d'assurance. La première source de données correspond à la liste des risques commerciaux d'une compagnie d'assurance. La seconde est le registre des entreprises du Québec (REQ), une source externe non structurée constituée d'informations textuelles sur des entreprises. Nous avons mis en correspondance les risques commerciaux avec les entités du REQ correspondantes selon les similarités calculées entre leurs noms et adresses respectives.

Cette problématique peut être formulée de la même manière qu'une classification binaire. C'est-à-dire qu'on peut utiliser un algorithme de classification permettant de déterminer si deux entités sont la même ou non. Toutefois, puisque nous cherchons à trouver la comparaison la plus probable parmi N entreprises possibles, nous n'utilisons que le score d'appartenance à la classe 1 (même entité). Cela nous permet d'utiliser des approches plus simples comme les algorithmes de similarité, qui sont différents des algorithmes de classification puisqu'ils permettent de déterminer la ressemblance entre deux chaînes de caractères plutôt que de les classer. Nous avons étudié neuf méthodes pour déterminer la ressemblance entre lesdites chaînes de caractères, de même que trois algorithmes de classification permettant de déterminer la ressemblance entre deux entités selon plus d'un attribut.

Premièrement, en utilisant les algorithmes de similarité sur la variante normalisée du nom sans mots outils, il est possible de coupler près de 66 % des risques commerciaux avec leur entité-vérité. Les méthodes de normalisation utilisées étant relativement rudimentaires, cela démontre qu'il n'y a que très peu de bruit dans les noms et qu'il est relativement facile de coupler la majorité des risques commerciaux. De plus, l'introduction d'un seuil minimal de similarité, qui filtre les associations ayant une trop faible similarité, a permis d'améliorer les résultats de couplage.

Deuxièmement, en utilisant les composantes de l'adresse plutôt que le nom, la proportion d'entreprises couplées est moindre. En effet, avec l'adresse, il n'est possible de coupler qu'au plus 48 % des entreprises. Les principaux problèmes relevés sont le positionnement des composantes de l'adresse et la présence d'abréviations. Par ailleurs, l'entité-vérité de plusieurs

risque commerciaux ne comporte pas d'adresse du fait de la politique de confidentialité du REQ. Il en résulte une sous-évaluation des résultats. Toutefois, même en retirant ces risques commerciaux, les résultats demeurent inférieurs à ceux obtenus en utilisant le nom.

Troisièmement, il a été démontré empiriquement que l'utilisation des algorithmes d'apprentissage automatique sur le nom et l'adresse conjointement, en l'occurrence avec le perceptron multicouche, permet de coupler près de 80 % des risques commerciaux sur un ensemble de validation. Ces résultats représentent une amélioration de 10 % par rapport à ceux de la meilleure configuration retenue jusque-là (Jaccard sur le nom seulement) pour le même ensemble de données.

Finalement, l'assouplissement de notre hypothèse initiale afin de pouvoir utiliser l'ensemble des N entités les plus probables d'être le doublon d'une entreprise nous a permis d'atteindre près de la plus haute valeur de rappel possible (93 %). Nous avons effectivement obtenu 91,07 % avec le perceptron multicouche. Il faut toutefois tenir compte du temps nécessaire à la génération des vecteurs d'informations, et à l'inférence des algorithmes d'apprentissage automatique. Effectivement, l'exécution de ces étapes prend environ 1,3 secondes par entreprise, contre 0,3 pour l'algorithme de Jaccard.

Travaux futurs

Il existe différentes pistes d'amélioration des travaux de ce mémoire. Premièrement, les plongements des mots pourrait permettre de générer des vecteurs d'informations de meilleure qualité pour les algorithmes de classification. Citons notamment les plongements de mots construits à partir du nom (Foxcroft *et al.*, 2019) ou à partir de l'entité au complet (Singh *et al.*, 2019; Wu *et al.*, 2017).

Deuxièmement, si l'on obtenait une représentation de chaque risque commercial, par exemple un plongement de mots, il serait possible d'utiliser un réseau siamois de détection de doublons. Ce type d'approche consiste à calculer la valeur absolue de la différence entre deux vecteurs d'informations pour ensuite l'utiliser comme source d'informations pour un algorithme d'apprentissage profond de classification (Imtiaz *et al.*, 2020; Godbole *et al.*, 2018).

Troisièmement, une solution pour améliorer les résultats du couplage par l'adresse serait d'utiliser des données géospatiales. À partir d'une telle représentation des adresses, il est possible de déterminer si ces dernières sont linéairement proches ou loin l'une de l'autre, et donc correspondantes ou non (Sehgal *et al.*, 2006). Cette piste de solution est intéressante puisque ce type de représentation peut être généré facilement à partir de l'adresse.

Finalement, les résultats indiquent que pour certains algorithmes de similarités (p. ex. Jaro-Winkler), le retrait des mots outils permet d'obtenir de meilleurs résultats. Toutefois, la liste de mots outils utilisée ici a été établie selon l'utilisation générale de la langue.

D'autres mots courants mais sans signification importante dans notre contexte pourraient donc y être ajoutés, notamment « inc », « enr » ou « associés ». Une étude plus poussée des mots outils spécifiques au domaine qui nous intéresse pourrait permettre d'en identifier et d'en ajouter davantage à la liste des mots outils devant être retirés lors du nettoyage des données. Une approche telle que *TF-IDF* (Sammut et Webb, 2010), qui accorde un poids à un mot relativement à son nombre d'occurrences dans un texte, pourrait permettre ce genre d'exploration.

Annexe A

Exemple de questionnaire client en assurance commerciale

Voici un exemple de questionnaire client [1] remis lors d'une demande de proposition d'assurance responsabilité civile des administrateurs et dirigeants pour des organismes à but non lucratif ou des associations. Ce questionnaire (voir plus loin) est un document typique servant à recueillir les informations nécessaires à la souscription d'une assurance pour un risque commercial.

Certaines questions sont relativement simples et pourraient être complétées à l'aide d'un système automatique. Par exemple, à partir du nom (Q.1), il serait intéressant de pouvoir remplir l'adresse postale de l'organisme (Q.3) ou encore la nature de ses activités (Q.4). Les informations ainsi recueillies automatiquement pourraient ensuite être validées par un représentant de l'entreprise au moment de répondre aux autres questions. En revanche, d'autres questions sont plus complexes et pourraient être difficilement complétées à l'aide de ce processus. Par exemple, le « pourcentage des services fournis ou des activités ayant lieu dans les [différents] pays » (Q.8), étant souvent privé, est une information presque impossible à trouver dans une source externe publique.

Toutefois, il pourrait être possible d'obtenir cette information sur certaines entreprises, comme celles qui sont inscrites sur les marchés financiers, puisque ces dernières sont obligées de la divulguer. Par contre, les informations de ce type ne sont pas accessibles pour la majorité des entreprises au Canada étant donné que 97,9 % (Statistique Canada, 2017) de celles-ci comptent moins de 100 employés et ne sont généralement pas des compagnies à actionnariat public.

En somme, cet exemple de questionnaire d'assurance permet de démontrer la pertinence de l'utilisation d'une source externe, puisqu'il est possible d'y trouver plusieurs réponses.

1. Le nom de la compagnie d'assurance émettant cette proposition a été volontairement retiré, car l'objectif n'est pas d'identifier la proposition d'une entreprise en particulier, mais uniquement d'illustrer le genre de question pouvant se retrouver dans ce type de questionnaire.

Si un contrat est émis, la couverture d'assurance sera limitée aux réclamations présentées pour la première fois contre l'assuré pendant la période d'assurance.

Veuillez joindre un exemplaire de chacun des documents suivants, afin de recevoir une soumission :

- Derniers états financiers vérifiés, mission d'examen ou avis au lecteur (si requis - voir section information financière)
- Certificat de constitution (si l'Organisme a été fondé récemment)
- Détails additionnels sur une feuille séparée lorsque requis ci-dessous

Proposant/Informations générales

1. Nom de l'Organisme formulant la présente demande d'assurance (si l'Organisme a des filiales, inscrire le Nom de l'Organisme mère seulement) :

   _____

2. Le dirigeant suivant est désigné comme représentant (de l'Organisme) dans le but de recevoir tous les avis de l'Assureur ou de ses représentants autorisés en rapport avec la présente assurance :

   Nom _____ Titre _____

3. Adresse postale de l'Organisme : _____

4. Nature des activités : _____

5. Site Web : _____

6. L'Organisme est en activité depuis : et, est constitué en vertu de la loi (inscrire la province, l'état ou le pays) :

   _____

7. Est-ce que l'Organisme a des filiales ou des sociétés affiliées pour lesquelles une assurance est requise? Oui ☐ Non ☐

Nom de/des (l')entité(s)	Activités/Services	% des droits de vote	À but lucratif	Constituée(s) en vertu de quelle loi?
		%	Oui ☐ Non ☐	
		%	Oui ☐ Non ☐	
		%	Oui ☐ Non ☐	

8. Pourcentage des services fournis ou des activités ayant lieu dans les pays suivants :

   Canada ____ % États-Unis ____ % Autre pays, veuillez spécifier ( ) ____ %

9. Est-ce que l'Organisme ou l'une de ses filiales est impliquée ou envisage une fusion, une consolidation, une acquisition, une cession d'activités ou la vente d'une portion de ses activités ou est-ce qu'une transaction similaire a été envisagée ou complétée au cours des trois dernières années? Oui ☐ Non ☐

10. Est-ce que l'Organisme impose ou recommande des sanctions disciplinaires résultant d'un examen de la pratique ou en lien avec l'établissement de normes de pratiques professionnelles de ses membres? Oui ☐ Non ☐

11. Est-ce que l'Organisme exerce des activités syndicales ou offre des services de négociation de conventions collectives? Oui ☐ Non ☐

12. Est-ce que l'Organisme offre des services professionnels? Oui ☐ Non ☐

13. Est-ce que l'Organisme participe à toute forme de recherche, développement ou expérimentation? Oui ☐ Non ☐

14. Est-ce que l'Organisme publie des magazines, périodiques ou bulletins autres qu'un rapport d'activités? Oui ☐ Non ☐

15. Est-ce que l'Organisme fait affaires avec des sociétés contrôlées par l'une des personnes qui seraient assurées par la présente assurance? Oui ☐ Non ☐

16. Est-ce que l'Organisme offre, fait la promotion, parraine ou fourni toute forme d'assurance pour ses membres ou non-membres? Oui ☐ Non ☐

Si une ou plusieurs questions ci-dessus sont répondues par « OUI », veuillez fournir tous les détails sur une feuille séparée.

Responsabilité des fiduciaires

Est-ce que l'Organisme fournit un régime de retraite? Oui ☐ Non ☐

Veuillez remplir les questions suivantes <u>seulement</u> si un régime de retraite est fourni par l'Organisme.

1. Nom(s) du(des) régime(s) de retraite : _____
2. Date de constitution : _____
3. Parrainage : Employeur unique ☐ Multi-employeur (négocié collectivement) ☐ Autre ☐ _____
4. Prestations déterminées ☐ Contributions déterminées ☐ Autre ☐ _____
5. Est-ce que le régime a déjà fait l'objet d'une conversion d'un régime à prestations déterminées vers un régime à cotisations déterminées? Oui ☐ Non ☐

   Si oui, fournir la date de conversion : _____
6. Est-ce que le régime est suffisamment capitalisé et solvable tel qu'attesté par une évaluation actuarielle? Oui ☐ Non ☐

   a) Actif total du régime (000 $) : _____ $

   b) Surplus/(perte) selon le dernier rapport actuariel : _____ $

   c) Nombre de participants : _____
7. Est-ce qu'il y a une politique de placement écrite convenue avec le conseiller en placements? Oui ☐ Non ☐
8. Est-ce qu'un régime a déjà été impliqué dans une transaction interdite? Oui ☐ Non ☐
9. Au cours de la dernière année, y a-t-il un (des) régime(s) qui a/ont cessé ses activités ou y a-t-il un régime qui prévoit cesser ses activités prochainement? Oui ☐ Non ☐
10. Y a-t-il des cotisations d'employeur en souffrance pour tout régime? Oui ☐ Non ☐

Assurance antérieure

Ne pas répondre si la présente proposition vise le renouvellement de votre contrat existant chez Intact Assurance.

Un Assureur a-t-il déjà décliné, refusé le renouvellement ou annulé un contrat d'assurance responsabilité des administrateurs et dirigeants, responsabilité en matière de pratiques d'emploi ou responsabilité des fiduciaires au proposant, au cours des trois dernières années? Oui ☐ Non ☐

	Administrateurs et dirigeants	Pratiques d'emploi	Fiduciaires
Montant de garantie	$	$	$
Franchise	$	$	$
Date d'expiration			

Historique des réclamations

Ne pas répondre si la présente proposition vise le renouvellement de votre contrat existant chez Intact Assurance.

Au cours des trois dernières années, un des administrateurs, dirigeants ou l'Organisme a-t-il été impliqué dans :

a) des procédures ou enquêtes civiles, criminelles, administratives or réglementaires? Oui ☐ Non

b) des réclamations produites aux termes d'un contrat d'assurance de la responsabilité des administrateurs et dirigeants, de responsabilité en matière de pratiques d'emploi ou de responsabilité des fiduciaires, ou un avis de réclamation éventuelle a-t-il été donné à l'Assureur? Oui ☐ Non ☐

c) des réclamations ou des réclamations potentielles ayant fait l'objet d'un avis écrit sous tout autre contrat d'assurance responsabilité? Oui ☐ Non ☐

d) des actions en justice collective, des recours collectifs ou des actions obliques? Oui ☐ Non ☐

Si une ou plusieurs questions ci-dessus sont répondues par « OUI », veuillez fournir tous les détails sur une feuille séparée.

Attestation de connaissance antérieure

La déclaration ci-dessous est requise de tous les proposants qui demandent une police d'assurance pour la première fois, l'ajout d' une nouvelle garantie qui ne fait pas partie de la police expirante, ou l'augmentation des montants de garantie plus élevés que ceux de la police expirante.

Ne pas remplir si la proposition vise le renouvellement d'une police existante avec Intact Assurance et si aucun changement dans la couverture ou aucune augmentation des montants de garantie n'est requis.

a) Les personnes ou sociétés à assurer sont-elles au courant de faits, circonstances, de situations, de transactions, d'évènements, d'actes, d'erreurs ou d'omissions qui sont susceptibles de donner lieu à une réclamation? Oui ☐ Non ☐

Dans l'affirmative, veuillez fournir tous les renseignements pertinents :

Il est entendu que, si une personne, un administrateur, un dirigeant, un Organisme, une filiale ou toute autre entité liée à assurer a connaissance de tels faits, circonstances, situations, transactions, évènements, actes, erreurs ou omissions qu'ils aient été déclarés ou non, toute réclamation ou poursuite qui en découlera sera exclue de la couverture accordée en vertu de toutes les polices souscrites par Intact Assurance.

Déclarations

Le soussigné désigné comme dirigeant de l'Organisme :

a) déclare qu'il est dûment autorisé par l'Organisme à remplir la présente proposition et les déclarations faites dans les présentes sont véridiques et complètes;

b) déclare que des efforts raisonnables ont été déployés pour obtenir des renseignements suffisants afin que le présent formulaire de proposition soit rempli de manière appropriée et exacte;

c) déclare que les états financiers soumis avec la présente proposition (le cas échéant) représentent la situation financière actuelle de l'Organisme, y compris ses filiales (dans le cas contraire, veuillez donner des détails sur une feuille distincte);

d) reconnaît que les informations, représentations et renseignements soumis seront utilisés par Intact Assurance et seront considérés comme la base de l'acceptation des risques assumés par l'Assureur en vertu de la présente assurance, si un contrat est émis;

e) accepte que si des modifications sont apportées aux renseignements qu'il a donné dans la présente proposition entre la date de celle-ci et la date d'entrée en vigueur du contrat, il en avisera immédiatement Intact Assurance par écrit et, sous toutes réserves et sans restreindre la portée de tout autre recours, Intact Assurance pourra révoquer ou modifier toute soumission de prix en vigueur, ainsi que toute autorisation ou entente visant à engager la garantie;

f) reconnaît que la présente proposition et tous documents qui y sont joints font partie du contrat, si un contrat d'assurance est émis.

Il est entendu que les renseignements, représentations et les documents supplémentaires joints à la présente proposition d'assurance sont véridiques et sont la base du contrat d'assurance. Les termes et conditions, incluant les montants de garantie, offerts par Intact Assurance peuvent différer de ceux demandés par le proposant. Il est de plus entendu que ce contrat d'assurance ne pourra pas être annulé ou ces protections exclues en conséquence d'une fausse déclaration contenue dans cette proposition, sauf pour l'Organisme, ses filiales et les personnes assurées ayant connaissance ou effectuant lesdites déclarations mensongères.

Signature	Fonction (Président, Directeur général, Chef du contentieux)
Date	Organisme

Annexe B

DeepTagger

Ce chapitre présente le marqueur séquentiel permettant d'étiqueter les composantes d'une adresse introduit dans le chapitre 4 et ayant été développer conjointement avec Marouane Yassine, étudiant au baccalauréat.

B.1 Étiqueter les composantes d'une adresse

Les composantes de l'adresse utilisées dans le chapitre 4 correspondent aux composantes d'une adresse regroupées par type d'étiquette, c'est-à-dire le regroupement des composantes du numéro civique, du nom de rue, etc. En regroupant par groupe d'étiquettes les composantes d'une adresse, nous sommes en mesure de comparer deux groupes d'étiquettes appartenant à la même catégorie. Toutefois, la segmentation d'une adresse par types d'étiquette n'est pas une tâche totalement résolue. En effet, les options de segmentation offertes sont soit dépendantes d'une source externe payante (Google APIs, 2020), conçues pour une région particulière (États-Unis (Busboom, 2014; Gregg *et al.*, 2014)) ou faiblement supportées dans le langage de programmation Python (Openvenues, 2016). Parce que nous tentons de développer une solution pouvant s'intégrer dans un processus déjà complexe, toute diminution des dépendances inutiles est favorable. En effet, la contrainte d'utiliser une source externe payante est non appréciable dans un contexte de recherche. De plus, parce que nous nous intéressons à des adresses multilingues canadiennes, les options de segmentation des adresses développées sur des données des États-Unis ne permettent pas d'obtenir de bons résultats sur les adresses francophones. Il est à noter qu'aucune option canadienne n'est disponible actuellement. Finalement, l'utilisation d'une librairie développée dans le langage de programmation C, mais partiellement supportée en Python n'est pas plus intéressante pour les contraintes de minimisation des dépendances. En plus, cette librairie ne supporte pas encore la segmentation des numéros d'unité, de bureau et d'appartement, composantes relativement présentent dans le jeu de données. Ce sont ces contraintes qui ont motivé la proposition d'une solution nécessitant peu de dépendance supplémentaire et permettant de segmenter l'ensemble des composantes

d'une adresse.

Trois catégories d'approches peuvent être utilisées pour résoudre la problématique d'étiquetage des composantes d'une adresse, notamment les approches à base de règles heuristiques, probabilistes et par apprentissage automatique. La première catégorie repose sur la conception de règles permettant l'extraction de l'information désirée. Deux méthodes sont utilisées pour concevoir ces règles, elles sont soit conçues par un expert du domaine (Churches *et al.*, 2003) ou à l'aide d'un système d'extraction de règles à partir de données non structurées (Califf et Mooney, 1997) ou semi-structurées (Soderland, 1999). Toutefois, comme Wang *et al.* (2016) l'ont démontré, ce type d'approche nécessite une expertise souvent poussée du domaine afin d'obtenir des résultats souvent très peu intéressants. Pour améliorer les résultats, des approches de la seconde catégorie ont été proposées, soit les approches probabilistes. Celles-ci reposent principalement sur l'utilisation d'un modèle de Markov caché (MMC) (Li *et al.*, 2014a) et de champ aléatoire conditionnel (Wang *et al.*, 2016). Toutefois, avec l'avènement de l'apprentissage profond, la troisième catégorie d'approches permet d'obtenir des résultats selon l'état de l'art actuel. Deux types de modèles se retrouvent dans cette catégorie, les réseaux de neurones pleinement connectés (Sharma *et al.*, 2018) et les réseaux récurrents (RNN) [1] (Mokhtari *et al.*, 2019). L'un des avantages des RNN est la possibilité de les appliquer sur des séquences de longueurs variables. Ainsi, si la séquence est de cinq composantes ou de huit composantes, il est possible d'utiliser le même RNN pour effectuer l'étiquetage.

Un réseau RNN permet en quelque sorte d'agir comme un réseau à mémoire ayant la capacité d'inférer sur l'information présente, mais aussi sur les évènements antérieurs. Au fur et à mesure que le réseau est appliqué sur une séquence, il met à jour l'information historique de son inférence de la séquence. Ce type de réseau correspond donc à une version plus complexe d'un MMC (Liu *et al.*, 2019). Toutefois, optimiser les réseaux RNN est parfois complexe parce qu'ils sont très sensibles au problème de disparition du gradient (Hochreiter et Schmidhuber, 1997a). Pour résoudre ce problème, nous utilisons des réseaux de neurones récurrents à mémoire court terme et long terme (LSTM) [2] (Hochreiter et Schmidhuber, 1997a). L'idée du LSTM est que chaque cellule récurrente est liée au même état caché h que le RNN, mais également à un état c de la cellule qui joue le rôle de mémoire. Le passage de c_{t-1} c_t se fait par un transfert à gain constant et égal à 1. Ainsi, les erreurs se propagent aux pas antérieurs sans phénomène de disparition du gradient (Hochreiter et Schmidhuber, 1997a). De plus, comme les RNN, il est possible d'utiliser les LSTM de manière bidirectionnelle (Schuster et Paliwal, 1997; Graves *et al.*, 2005). Le terme bidirectionnel signifie que le réseau est appliqué sur la séquence dans les deux sens sans partager l'information entre les sens. Par exemple, la Figure B.1 présente un réseau LSTM bidirectionnel appliqué sur une séquence x de longueur n. On observe que le LSTM directionnel de la gauche vers la

1. De l'anglais recurrent neural networks.
2. De l'anglais long short-term memory.

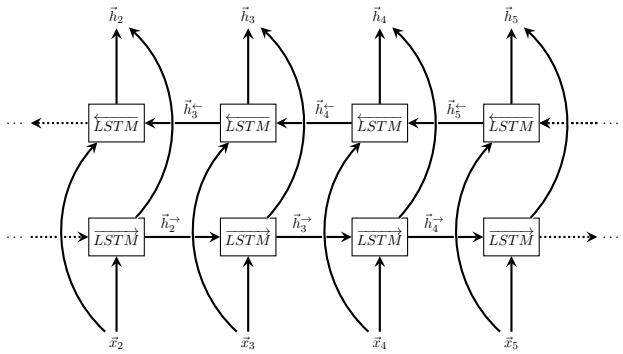

FIGURE B.1 – Représentation d'un LSTM bidirectionnel

droite (en bas) est appliqué sur la séquence des jetons x_0 à x_n alors que le LSTM directionnel de la droite vers la gauche (en haut) est appliqué sur la séquence de jetons x_n à x_0. Ce qui permet d'obtenir une représentation de la droite vers la gauche de la séquence ainsi qu'une seconde représentation de la gauche vers la droite de la séquence. Autrement dit $\vec{h}_2^{\rightarrow}$ est une matrice de 2 par n où la première rangée (1) correspond à la représentation dans le « bon » sens et la seconde rangée (2) correspond à la représentation de la séquence à l'envers.

Il est aussi possible d'utiliser un modèle de séquence à séquence, une architecture d'encodeur-décodeur composée de plusieurs LSTM (Hochreiter et Schmidhuber, 1997a). Cette approche permet d'utiliser la représentation de plusieurs LSTM, le vecteur de contexte, comme information d'entrée à un réseau de plusieurs autres LSTM permettant de décoder la séquence. La Figure B.2 illustre un exemple d'architecture qui utilise deux LSTM pour l'encodeur (en rouge), deux LSTM pour le décodeur (bleu) et une couche pleinement connectée (jaune) pour l'étiquetage. Le décodeur (bleu) utilise un premier LSTM (bas) pour encoder le mot, un second LSTM (haut) qui utilise la représentation de l'encodeur (rouge) et l'information du premier LSTM de l'encodeur. Mokhtari *et al.* (2019) ont utilisé cette approche pour étiqueter les adresses de requêtes dans des recherches cartographiques (p. ex. Google Maps). Ils rapportent une précision, avec leur architecture Seq2Seq, de 99,17 %. Toutefois, ce type d'architecture est plus complexe et souvent plus difficile à optimiser. Alors, pour ces contraintes, nous n'avons pas envisagé cette option.

Les données

Pour entraîner le modèle, nous utilisons un jeu de données d'adresse provenant de deux sources. La première correspond aux données déjà annotées du jeu de données privé, soit l'adresse de 21 443 risques commerciaux. Le jeu de données contient l'adresse non segmentée ainsi que cette dernière segmentée en ses composantes avec l'une des sept étiquettes. Le nom et la signification de chaque étiquette utilisée dans le jeu de données sont présentés au Tableau B.1. De plus, 2 226 données provenant du REQ ont aussi été étiquetées manuellement pour augmenter la

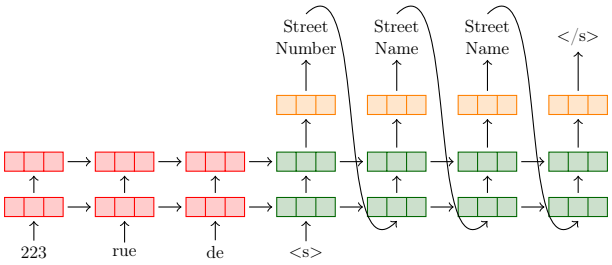

FIGURE B.2 – Architecture d'un modèle de séquence à séquence pour l'étiquetage. Les boîtes en rouges sont les LSTM encodeurs et celles en vertes sont les LSTM décodeurs

taille du jeu de données. En somme, le jeu de données complet contient 23 669 données.

Étiquette	Description	Exemple
StreetNumber	Numéro d'identification d'un édifice sur une rue	15
StreetName	Nom spécifique d'une rue dans une ville	Avenue Myrand
Orientation	Directions cardinales et intercardinales	Ouest
Unit	Identification d'un espace dans un édifice	1b
Municipality	Zone relativement peupler et organiser	Québec
Province	Division administrative d'un pays	Québec
PostalCode	Une combinaison de nombres et de lettres	G1V 0A7

Tableau B.1 – Liste des étiquettes, leur description et un exemple

Pour entraîner le modèle, nous utilisons 80 % des données pour l'entraînement du modèle, 10 % pour la validation de ce dernier après chaque itération d'entraînement et 10 % pour tester uniquement à la fin de l'entraînement.

Le modèle

Le marqueur séquentiel proposé, DeepTagger, correspond à un LSTM bidirectionnel qui utilise le plongement de mots fastText (Bojanowski *et al.*, 2016) en 300 dimensions pour encoder l'adresse ainsi qu'une couche linéaire pleinement connectée avec dimension en entrée de 600 (ce qui correspond aux deux sorties du LSTM bidirectionnel) et avec une dimension en sortie de taille 7.

Pour encoder la représentation des jetons de la séquence x, nous utilisons un plongement de mots. Ce type d'encodage correspond à une représentation vectorielle en n dimensions des mots d'un dictionnaire. La première intuition d'un plongement de mots est qu'il existe une relation linéaire entre les mots. Par exemple, la relation linéaire entre « roi » et « reine » est le genre, alors la relation de ces mots dans un espace vectoriel est que si l'on retire le mot « homme » à roi et que l'on ajoute le mot « femme », on doit obtenir « reine ». La

seconde est que les mots communs entre eux devraient être des voisins dans un espace vectoriel. Par exemple, chat et chats ou chien et chiens doivent être des voisins proches dans un espace vectoriel. Pour atteindre cette représentation, plusieurs algorithmes ont été proposés pour construire ces représentations vectorielles des mots entre eux, notamment Word2Vec (Mikolov *et al.*, 2013a) et Glove (Pennington *et al.*, 2014a). Par contre, parce que les algorithmes de Word2Vec et Glove utilisent uniquement le mot et son contexte, il est impossible d'obtenir une représentation vectorielle pour un mot n'ayant jamais été vu dans le lexique (OOV) [3]. Pour résoudre cette problématique Bojanowski *et al.* (2016) ont proposé d'utiliser les sous-séquences (Jurafsky et Martin, 2009a) de mots dans la construction du plongement de mots. Une sous-séquence d'un mot correspond aux N caractères consécutifs du mot. Par exemple, la division en sous-séquences de 3 caractères (tri-gramme) du mot « université » permet d'obtenir les tri-grammes suivant : <s><s>u, <s>un, uni, ..., é</s></s>, où <s> et </s> sont des caractères de début et de fin de la séquence permettant de remplir la séquence pour ne par perdre de caractères. L'utilisation des N-grams de caractères dans la construction du plongement de mots permet de construire un vecteur pour un mot inconnu à partir des N-grams du mot. En somme, nous utilisons les plongement de mots de fastText comme source d'encodage des mots de l'adresse pour le LSTM afin de résoudre les OOV plus facilement.

Finalement, nous utilisons l'entropie croisée comme fonction de perte ainsi que la méthode d'optimisation par descente du gradient stochastique (SGD) avec un taux d'apprentissage de 0,1 pour optimiser le réseau. Ce réseau est entraîné durant 50 itérations avec une taille de lot d'entraînement de 128 données et une amorce de départ de 42.

B.2 Résultats et analyse

Nos résultats sont comparés avec ceux de l'état de l'art présenté par Mokhtari *et al.* (2019), soit une précision de 99,17 %. On observe au Tableau B.2 que nos résultats sont très similaires à ceux de l'architecture Seq2Seq, la précision étant de 0,15 % inférieure à celle de l'état de l'art. Malgré la simplicité de notre modèle, nous sommes en mesure d'obtenir d'excellents résultats.

(%)	DeepTagger	Seq2Seq
Précision	99,02	99,17

Tableau B.2 – Résultats du modèle DeepTagger et Seq2Seq (Mokhtari *et al.*, 2019)

De plus, on observe au Tableau B.3 que la précision par étiquette est tout aussi bonne. Toutefois, le modèle éprouve de la difficulté à bien étiqueter les numéros civiques et les noms de rues. Cela est probablement dû à la complexité de certains cas, notamment lorsque le nom de la rue est un numéro (p. ex. 1re avenue). Une piste de solution pour améliorer les résultats dans cette situation serait d'utiliser un plus grand nombre de données.

3. De l'anglais *out-of-vocabulary*.

Étiquette	Présicion (%)	Étiquette	Précision (%)
StreetNumber	98,77	StreetName	97,29
Orientation	99,74	Unit	98,73
Municipality	97,50	Province	99,78
PostalCode	99,70	Unknown	99,91

Tableau B.3 – Résultats du modèle DeepTagger par étiquette

B.2.1 Discussion sur les résultats

Les résultats indiquent qu'il est possible d'obtenir des résultats très proches de l'état de l'art sans nécessiter une architecture complexe. Notre modèle étant composé uniquement d'un seul LSTM alors que le second modèle en utilise cinq, il semble que nos bons résultats soient principalement dûs à la qualité des données. Puisque nous n'avons pas utilisé le même jeu de données. De plus, un point important à mentionner est que les auteurs utilisent 15 types d'étiquettes, ce qui augmente de manière importante la granularité de l'étiquetage. Il est fort possible que cet différence dans les étiquettes ait un effet important sur nos résultats. En effet, on observe au Tableau B.3 que les étiquettes les plus difficiles à identifier sont, en ordre, *StreetName*, *Orientation*, *Unit* et *StreetNumber* et que les étiquettes suppémentaires par rapport à Mokhtari *et al.* (2019) sont des étiquettes avec une complexité similaire. Notamment, *Sub address Type* (édifice), *Building Name*, *Street Type* (boulevard, rue) et *Neighborhood*, *Separator* (proche de, dans). Toutefois, étant donné les bons résultats obtenus sur les données utilisées, une analyse plus profonde n'a pas été effectuée. Par contre, il est à noter que nous sommes en processus de développement d'une solution plus performante et multilingue. Mais au moment d'écrire ce mémoire, les résultats ne sont pas disponibles.

B.3 Conclusion

Pour conclure, le modèle proposé pour étiqueter les composantes d'une adresse permet d'obtenir des résultats très similaires à l'état de l'art tout en étant beaucoup plus simple que les modèles précédents.

Annexe C

Leveraging Subword Embeddings for Multinational Address Parsing

Dans l'annexe B, nous avons décrit le marqueur séquentiel permettant d'étiqueter les composantes d'une adresse créé et utilisé pour ce mémoire. La création de cet outil a mené à l'écriture d'un article où nous en proposons une version internationale. Cette dernière permet d'étiqueter des adresses de 20 pays en obtenant des résultats de l'état de l'art. De plus, dans cet article, nous explorons aussi l'efficacité de ce marqueur sur l'étiquetage d'adresses provenant de pays n'ayant pas été vus durant l'entraînement (*zero-shot transfert*). L'article a été accepté à la « *4th IEEE Conference on "Machine Learning and Natural Language Processing : Models, Systems, Data and Applications"* ».

Leveraging Subword Embeddings for Multinational Address Parsing

Marouane Yassine, David Beauchemin, François Laviolette, and Luc Lamontagne

Department of Computer Science and Software Engineering, Laval University
Group for Research in Artificial Intelligence of Laval University (GRAIL)
Québec, Canada

Résumé

Address parsing consists of identifying the segments that make up an address such as a street name or a postal code. Because of its importance for tasks like record linkage, address parsing has been approached with many techniques. Neural network methods defined a new state-of-the-art for address parsing. While this approach yielded notable results, previous work has only focused on applying neural networks to achieve address parsing of addresses from one source country. We propose an approach in which we employ subword embeddings and a Recurrent Neural Network architecture to build a single model capable of learning to parse addresses from multiple countries at the same time while taking into account the difference in languages and address formatting systems. We achieved accuracies around 99% on the countries used for training with no pre-processing nor post-processing needed. We explore the possibility of transferring the address parsing knowledge obtained by training on some countries' addresses to others with no further training in a zero-shot transfer learning setting. We achieve good results for 80% of the countries (33 out of 41), almost 50% of which (20 out of 41) is near state-of-the-art performance. In addition, we propose an open-source Python implementation of our trained models [1].

C.1 Introduction

Address Parsing is the task of decomposing an address into the different components it is made of. This task is an essential part of many applications, such as geocoding and record

1. https://github.com/GRAAL-Research/deepparse

linkage. Indeed, to find a particular location based on textual data, it is quite useful to detect the different parts of an address to make an informed decision. Similarly, comparing two addresses to decide whether two or more database entries refer to the same entity can prove to be quite difficult and prone to errors if based on methods such as edit distance algorithms given the various address writing standards.

There have been many efforts to solve the address parsing problem. From rule-based techniques (Xu *et al.*, 2012) to probabilistic approaches and neural network models (Abid *et al.*, 2018), a lot of progress has been made in reaching an accurate segmentation of addresses. These previous pieces of work did a remarkable job at finding solutions for the challenges related to the address parsing task. However, most of these approaches either do not take into account parsing addresses from different countries or do so but at the cost of a considerable amount of meta-data and substantial data pre-processing pipelines (Mokhtari *et al.*; Li *et al.*, 2014b; Wang *et al.*, 2016; Sharma *et al.*, 2018).

Our work comes with two objectives. Firstly, we propose an approach for multinational address parsing using a Recurrent Neural Network (RNN) architecture. We start by addressing the multilingual aspect of the problem by employing multilingual sub-word units. Then we train an architecture composed of an embedding layer followed by a sequence-to-sequence (Seq2Seq) model. Secondly, we evaluate the degree to which a model trained on countries' addresses data can perform well at parsing addresses from other countries.

C.2 Related work

Since address parsing is a sequence tagging task, it has been tackled using probabilistic methods mainly based on Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Li *et al.*, 2014b; Wang *et al.*, 2016; Abid *et al.*, 2018). For instance, Li *et al.* (2014b) proposed a large scale HMM-based parsing technique capable of segmenting a large number of addresses, whilst being robust to possible irregularities in the input data. In addition, Wang *et al.* (2016) implemented a discriminative model using a linear-chain CRF coupled with a learned Stochastic Regular Grammar (SRG). This approach allowed the authors to better address the complexity of the features while capturing higher-level dependencies by applying the SRG on the CRF outputs as a score function, thus taking into account the possible lack of features for a particular token in a lexicon-based model. These probabilistic methods usually rely on structured data as well as some sort of prior knowledge of this data for feature extraction or in order to implement algorithms such as Viterbi (Viterbi, 1967), especially in the case of generative methods.

In recent years, new methods (Mokhtari *et al.*; Abid *et al.*, 2018) utilizing the power of neural networks have been proposed as solutions for the address parsing problem. Using a single hidden layer feed-forward model, Sharma *et al.* (2018) achieved state-of-the-art performance.

Their approach, however, relied on a pipeline of pre-processing and post-processing so as to deal with the different structures of address writing, as well as the possible prediction errors. For instance, the input data is normalized to reduce noise and to standardize the many variations that can refer to the same word, such as *road* and *rd*. In addition, the model's predictions are put through a rule-based validation step to make sure that they fit known patterns. In contrast, Mokhtari *et al.* proposed a deep learning approach based on the use of RNN. Their experiments focused on comparing the performance of both unidirectional and bidirectional vanilla RNN and Long-Short Term Memory Models (LSTM) (Hochreiter et Schmidhuber, 1997b), as well as a Seq2Seq. The models achieved high accuracy on test sets with the Seq2Seq leading the scoreboard on most of them with no particular pre-processing needed during the inference process.

Note however that despite reaching notable performances, the aforementioned approaches are limited to parsing addresses from a single country and would need to be adjusted to support a multinational scope of address parsing. To tackle this problem, Libpostal [2], a library for international address parsing, has been proposed. This library uses a CRF-based model trained with an averaged Perceptron for scalability. The model was trained on data from each country in the world and was able to achieve a 99.45% full parse accuracy [3]. However, this requires putting addresses through a heavy pre-processing pipeline before feeding them to the prediction model. It is our understanding that no neural network approaches were proposed for multinational address parsing with a single model. This work aims at building a single model solution capable of parsing addresses from multiple countries, as well as exploring the possibility of zero-shot transfer from some countries addresses to others'.

C.3 Subword embeddings

The use of subword embeddings has become popular across Natural Language Processing tasks given the performance enhancements they provide to neural network models. Word embeddings (Mikolov *et al.*, 2013b; Pennington *et al.*, 2014b) are usually augmented by character-level or subword-level information before being fed to the model as inputs, thus granting it a more meaningful representation of words. This strategy is employed by the word embeddings library fastText (Bojanowski *et al.*, 2016) in which a representation of words as character n-grams is used along with words representations in order to produce embeddings. This approach allows for a model capable of producing richer embeddings, as well as embeddings for out-of-vocabulary words (OOV), which are computed as the sum of their n-gram fractions' embeddings. For example, the embedding of the OOV word "H1A 1B1" using a bi-gram is the sum of the fractions' embedding of {H1, 1A, A1, 1B, B1}.

2. https://github.com/openvenues/libpostal
3. The accuracy was computed considering the entire sequence and was not focused on individual tokens.

C.3.1 Byte-pair Encoding

Byte-pair encoding (BPE) (Gage, 1994) is a data compression algorithm which iteratively replaces the most frequent occurrences of adjacent bytes with a new set of bytes to find a more compact representation of the said data. A new approach for word segmentation based on the BPE algorithm was introduced by Sennrich *et al.* (2016). Their technique, which was proposed to solve the OOV problem in Neural Machine Translation (NMT), consists of representing text as a sequence of characters that are iteratively merged using the same reasoning behind BPE. This approach paved the way for the authors to address NMT with an open-vocabulary solution. Another application of BPE is BPEmb (Heinzerling et Strube, 2017), a set of embedding models that were trained to produce subword embeddings based on a BPE decomposition of text. BPEmb offers pre-trained models on 275 languages, as well as MultiBPEmb, which is a single model trained on the shared vocabulary of the 275 languages. These models were shown to have a performance similar to other subword embedding techniques on an entity typing task while outperforming these techniques on some languages.

C.4 Architecture

The following section describes the architecture of our model, which is composed of an embedding model (subsection C.4.1) and a tagging model (subsection C.4.2) as shown in Figure C.1.

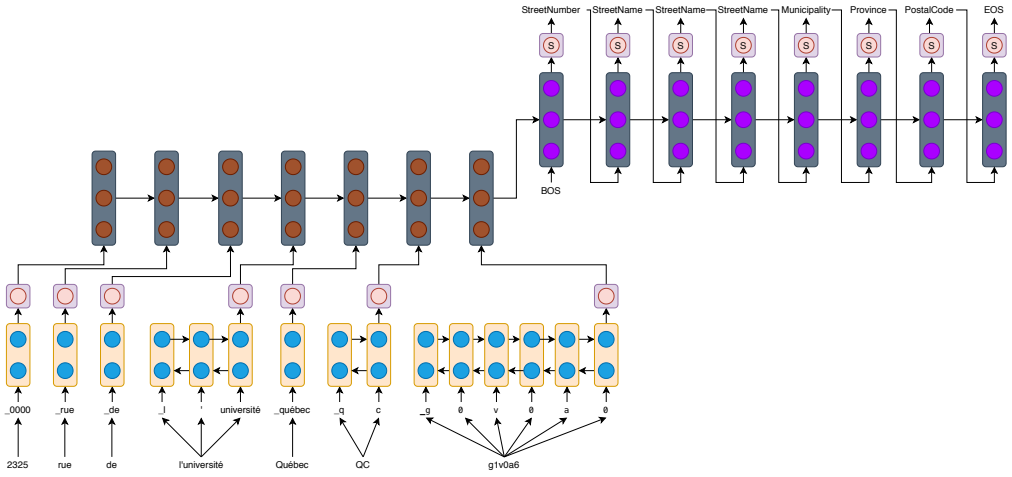

FIGURE C.1 – Illustration of our architecture using the BPEmb embedding model. Each word in the address is encoded using MultiBPEmb (the BPE segmentation algorithm replaces the numbers in the address by zeros). The subword embeddings are fed to the BiLSTM (rounded rectangle with two circles). The last hidden state for each word is run through a fully connected layer (rounded rectangle with one circle). The resulting embeddings are given as input to the Seq2Seq (rounded rectangle with three circles). The 'S' in the fully connected layer following the Seq2Seq decoder stands for the Softmax function.

C.4.1 Embedding Model

Since our main objective is to build a single neural network for parsing addresses from multiple countries, it is necessary to have access to embeddings for different languages at runtime. Some libraries, such as fastText (Joulin *et al.*, 2018) and MUSE (Conneau *et al.*, 2017), offer alignment vectors that enable the projection of word embeddings from different languages in the same space. However, these techniques would require detecting the source language as well as specifying the target language to use the proper alignments, which we consider an unnecessary overhead for the task at hand. To resolve the embedding issue, we propose the following two methods.

First, we use a fixed pre-trained monolingual fastText model (pre-trained on the French language) (**fastText**). We chose French embeddings since the French language shares Latin roots with many languages in our test set. It is also due to the considerable size of the corpus on which these embeddings were trained.

Second, we use an encoding of words using MultiBPEmb and merge the obtained embeddings for each word into one word embedding using a RNN. This method has been shown to give good results in a multilingual setting (Heinzerling et Strube, 2019). Our RNN network of choice is a Bidirectional LSTM (Bi-LSTM) with a hidden state dimension of 300. We build the word embeddings by running the concatenated forward and backward hidden states corresponding to the last time step for each word decomposition through a fully connected layer of which the number of neurons is equal to the dimension of the hidden states. This approach produces a 300-dimensional word embeddings. We refer to this embeddings model technique as **BPEmb**.

We run a comparison of the two methods (**fastText** and **BPEmb**) to evaluate which one gives better results in our setting.

C.4.2 Tagging Model

Our downstream tagging model is a Seq2Seq model consisting of a one-layer unidirectional LSTM encoder and a one-layer unidirectional LSTM decoder followed by a fully-connected linear layer with a softmax activation. Both the encoder's and decoder's hidden states are of dimension 1024. The embedded address sequence is fed to the encoder that produces hidden states, the last of which is used as a context vector to initialize the decoder's hidden states. The decoder is then given a Beginning Of Sequence (BOS) token as input, and at each time step, the prediction from the last step is used as input. To better adapt the model to the task in hand and to facilitate the convergence process, we only require the decoder to produce a sequence with the same length as the input address. This approach differs from the traditional Seq2Seq architecture in which the decoder makes predictions until it predicts the EOS token. The decoder's outputs are forwarded to the linear layer of which the number of neurons is equal to the tag space dimensionality. The softmax activation function computes probabilities

over the linear layer's outputs to predict the most likely token at each time step.

C.5 Data

Our dataset was built using the open-source data on which Libpostal's models were trained and of which we have collected the address data of 61 countries. Twenty countries were used for multinational training with a sample size of 100,000 addresses per country while the rest of the samples was left out as holdout for testing. The other countries' data was also left for zero-shot transfer evaluation. Tables C.1 and C.2 show the number of samples per country in both test sets ordered by number of examples per country. The color in the table will be discussed later on.

We introduce eight tags, namely StreetNumber, StreetName, Unit, Municipality, Province, PostalCode, Orientation, and GeneralDelivery, as opposed to Libpostal, which utilizes 20 tags. This was motivated by the common presence of the chosen tags in most of the countries that are included in our datasets. Also, it is not guaranteed that all addresses contain each tag category's elements since some addresses might not contain elements of some tag categories. Figure C.2 shows address samples for different countries with the corresponding tags. Each color represents one of the five different patterns present in our dataset (Rhind, 2020). We also find that some countries' address format is composed of different patterns (e. g. Belarus that use the second and fifth patterns). No color is used for these countries.

Tableau C.1 – Number of samples per country in the holdout test set for training countries

Country	Number of samples	Country	Number of samples	Country	Number of samples	Country	Number of samples
United States	8,000,000	Germany	1,576,059	Poland	459,522	Czechia	195,269
Brazil	8,000,000	Spain	1,395,758	Norway	405,649	Italy	178,848
South Korea	6,048,106	Netherlands	1,202,173	Austria	335,800	France	20,050
Australia	5,428,043	Canada	910,891	Finland	280,219	United Kingdom	14,338
Mexico	4,853,349	Switzerland	474,240	Denmark	199,694	Russia	8115

Tableau C.2 – Number of samples per country in the zero-shot test set

Country	Number of samples	Country	Number of samples	Country	Number of samples	Country	Number of samples
Belgium	66,182	Slovenia	9773	Réunion	2514	Bangladesh	888
Sweden	32,291	Ukraine	9554	Moldova	2376	Paraguay	839
Argentina	27,692	Belarus	7590	Indonesia	2259	Bosnia	681
India	26,084	Serbia	6792	Bermuda	2065	Cyprus	836
Romania	19,420	Croatia	5671	Malaysia	2043	Ireland	638
Slovakia	18,975	Greece	4974	South Africa	1388	Algeria	601
Hungary	17,460	New Zealand	4678	Latvia	1325	Colombia	569
Japan	14,089	Portugal	4637	Kazakhstan	1087	Uzbekistan	505
Iceland	13,617	Bulgaria	3716	New Caledonia	1036		
Venezuela	10,696	Lithuania	3126	Estonia	1024		
Philippines	10,471	Faroe Islands	2982	Singapore	968		

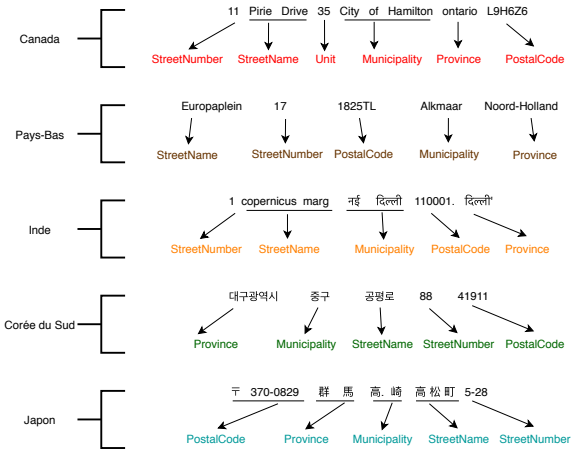

FIGURE C.2 – Address samples per country

C.6 Experiments

For our experiments, we trained our two models (**fastText** and **BPEmb**) five times each [4] for 200 epochs with a batch size of 2048. An early stopping with a patience of fifteen epochs was also applied during training. We initialize the learning rate at 0.1 and use learning rate scheduling to lower it by a factor of 0.1 after ten epochs without loss reduction. Our loss function of choice is the Cross-Entropy loss due to its suitability for the softmax function. The optimization is done through Stochastic Gradient Descent.

Also, to speed up the convergence, we use teacher forcing (Williams et Zipser, 1989), a method that consists of using the ground truth instead of the previous time step's prediction as input for the decoder during training. We do so by randomly sampling part of the training data at runtime. 80% of the training datasets were used to train the models, and 20% was kept for validation. The architecture, as well as the training of the models, were implemented using Pytorch (Paszke *et al.*, 2019) and Poutyne (Paradis, 2018).

C.6.1 Evaluation Procedure

We train our two models on our multinational dataset, the difference between the models being the word embedding method employed (**fastText** and **BPEmb**). Each model has been trained five times and we report the models' mean accuracy and standard deviation on the per-country holdout data in Table C.3. The accuracy for each sequence is computed as the proportion of the tags predicted correctly by the model. As such, predicting all the tags for a sequence correctly yields a perfect accuracy. More precisely, errors in tag predictions have an

4. Using each of the following seeds $\{5, 10, 15, 20, 25\}$. When a model didn't converge (a high loss value on train and validation), we retrain the model using a different seed (30).

impact on the accuracy for a given sequence. However, the accuracy will not be null unless all the predicted tags for the sequence are incorrect. Our two models mean results are compared to those of Libpostal evaluated on the same validation dataset. However, since Libpostal's preprocessing step has introduced an incompatibility between the input data and the ground truth, we were obliged to discard a few addresses (less than 0.3 % of them) during the scores' generation for some countries (South Korea, Germany, Norway, Poland, Serbia, Japon and Bulgaria).

Tableau C.3 – Multinational models' mean accuracy (and standard deviation) on holdout datasets for training countries

Country	Libpostal	FastText	BPEmb	Country	Libpostal	FastText	BPEmb
United States	**99.95**	99.61 ± 0.09	99.67 ± 0.09	Poland	**99.93**	99.69 ± 0.07	99.89 ± 0.04
Brazil	**99.94**	99.40 ± 0.10	99.42 ± 0.15	Norway	**99.96**	99.46 ± 0.06	98.41 ± 0.63
South Korea	77.25	99.96 ± 0.01	$\mathbf{100.00 \pm 0.00}$	Austria	**99.38**	99.28 ± 0.03	98.98 ± 0.22
Australia	98.86	99.68 ± 0.05	$\mathbf{99.80 \pm 0.05}$	Finland	99.84	99.77 ± 0.03	$\mathbf{99.87 \pm 0.01}$
Mexico	99.62	99.60 ± 0.06	$\mathbf{99.68 \pm 0.06}$	Denmark	**99.96**	99.71 ± 0.07	99.90 ± 0.03
Germany	99.82	99.77 ± 0.04	$\mathbf{99.89 \pm 0.03}$	Czechia	99.83	99.57 ± 0.09	$\mathbf{99.89 \pm 0.04}$
Spain	**99.89**	99.75 ± 0.05	99.85 ± 0.04	Italy	99.66	99.73 ± 0.05	$\mathbf{99.81 \pm 0.05}$
Netherlands	**99.96**	99.61 ± 0.07	99.88 ± 0.03	France	**99.84**	99.66 ± 0.08	99.69 ± 0.11
Canada	**99.94**	99.79 ± 0.05	99.87 ± 0.04	United Kingdom	99.28	99.61 ± 0.10	$\mathbf{99.74 \pm 0.08}$
Switzerland	99.58	99.53 ± 0.09	$\mathbf{99.75 \pm 0.08}$	Russia	**99.95**	99.03 ± 0.24	99.67 ± 0.11

Tableau C.4 – Multinational models' z-test significance test on holdout datasets for training countries (**bold** value are rejected null hypothesis with $\alpha = 0.001$)

Country	Libpostal FastText	Libpostal BPEmb	FastText BPEmb	Country	Libpostal FastText	Libpostal BPEmb	FastText BPEmb
United States	144.58	127.52	-20.56	Poland	26.86	7.19	-20.73
Brazil	185.68	180.75	-6.12	Norway	41.96	77.70	46.01
South Korea	-1243.10	-1246.13	-48.29	Austria	5.31	18.33	13.11
Australia	-157.17	-190.27	-41.23	Finland	5.46	**-3.15**	-8.56
Mexico	5.36	-17.76	-23.10	Denmark	19.02	6.49	-13.70
Germany	9.04	-17.43	-26.18	Czechia	14.87	-5.27	-19.50
Spain	27.96	10.05	-18.32	Italy	-3.51	-8.28	-4.83
Netherlands	58.93	21.80	-42.13	France	3.54	**3.11**	**-0.45**
Canada	28.50	16.08	-13.38	United Kingdom	-3.72	-5.55	**-1.95**
Switzerland	3.85	-14.05	-17.80	Russia	8.17	3.99	-5.10

C.6.2 Multinational Evaluation

First, we find that **BPEmb** gives better results than Libpostal for 9 of 20 countries, even if we have trained on just 100,000 examples of each country compared to near millions per country for Lipbostal. These results show that our approach can achieve similar results whilst requiring less data and without the need of pre-processing nor post-processing.

Second, we find the model using BPEmb embeddings to have the best performance across the board without considering Libpostal. The **BPEmb** model achieves better results than

fastText in most cases (all except Norway and Austria). We find that South Korea is the only country where a perfect accuracy was achieved using **BPEmb** (for four seeds out of five). Since South Korea is the only country using a different pattern in the training set where the province and municipality occur before the street name, it seems that our models might have memorized this particular pattern. To validate this intuition we randomly reordered 6000 South Korean addresses to follow either the first (red) or the second (brown) address pattern (equally divided between the two). We observe, after this reordering, that the mean accuracy drops to 28.04% considering that using a random tags annotation, we get a 12.29 % accuracy.

It was also interesting to notice the model accuracy is good when using fastText monolingual word embeddings, especially on South Korean addresses despite the entirely different alphabet. These results illustrate that our model, regardless of the embeddings model, learned the representation of an address sequence even if the words' representations are not native to the language (French vs Korean).

Overall, all our models achieve state-of-the-art performance on our dataset while using less data than previous approaches. However, **BPEmb** generates better results than **fastText** and similar ones to Libpostal. To further assess' models' performance, we report the models' z-test significance test in Table C.4. Our z-test null hypothesis is that the pair of models have equal performances, meaning that values smaller or greater than $|3.290527|$ allow us to reject the null hypothesis with $\alpha = 0.001$. A positive value means that the first model (left) has a significantly better performance than the second (right), and a negative value means the opposite. We observe that **BPEmb** has a significantly better performance for 8 out of 20 countries compared to Libpostal, which means that the performance between the two is quite similar. Besides, we observe that **BPEmb** has almost always significantly better performance than **fastText**. Considering these results, we can conclude that **BPEmb** is better than **fastText** and **BPEmb** is similar to Libpostal but using less training data and no pre-processing nor post-processing.

C.6.3 Zero-shot Evaluation

Since training a deep learning model to parse addresses from every country in the world would require a significant amount of data and resources, our ongoing work aims at achieving domain adaptation to be able to train on a reasonable amount of data and generalize to data from different sources. We begin by exploring how well our architecture can generalize in a zero-shot manner. To this end, we test each of our five trained models using the two embedding settings on address data from countries not seen during the training. The results are reported in Table C.5 ordered by dataset size. We choose not to bold Libpostal scores when they are the best since their model was trained on these addresses, making the comparison unfair to our models.

Tableau C.5 – Zero-shot transfer models' mean accuracy (and standard deviation) per country

Country	Libpostal	FastText	BPEmb	Country	Libpostal	FastText	BPEmb
Belgium	99.77	**88.14 ± 1.04**	87.29 ± 1.40	Faroe Islands	99.99	74.14 ± 1.83	**85.50 ± 0.11**
Sweden	99.96	81.59 ± 4.53	**90.76 ± 3.03**	Réunion	89.15	**96.80 ± 0.45**	93.67 ± 0.26
Argentina	99.34	86.26 ± 0.47	**88.04 ± 0.83**	Moldova	99.76	**90.18 ± 0.79**	86.89 ± 3.01
India	97.27	69.09 ± 1.74	**80.04 ± 3.24**	Indonesia	98.40	64.31 ± 0.84	**70.28 ± 1.64**
Romania	99.85	**94.49 ± 1.52**	91.65 ± 1.21	Bermuda	98.94	92.31 ± 0.60	**93.70 ± 0.35**
Slovakia	99.81	82.10 ± 0.98	**90.31 ± 3.88**	Malaysia	97.98	78.93 ± 3.78	**94.16 ± 0.49**
Hungary	99.69	**48.92 ± 3.59**	25.51 ± 2.60	South Africa	99.84	95.31 ± 1.68	**96.87 ± 0.96**
Japan	92.92	**41.41 ± 3.21**	35.33 ± 1.28	Latvia	99.36	**93.66 ± 0.64**	74.78 ± 4.33
Iceland	99.84	96.55 ± 1.20	**97.38 ± 1.18**	Kazakhstan	99.92	86.33 ± 3.06	**94.12 ± 1.94**
Venezuala	99.60	**94.87 ± 0.53**	93.05 ± 2.02	New Caledonia	99.89	**99.48 ± 0.15**	99.25 ± 0.19
Philippines	99.59	77.76 ± 3.97	**81.95 ± 8.07**	Estonia	99.98	**87.08 ± 1.89**	77.30 ± 1.22
Slovenia	99.88	95.37 ± 0.23	**97.47 ± 0.45**	Singapore	99.46	86.42 ± 2.36	**86.87 ± 2.01**
Ukraine	99.90	**92.99 ± 0.70**	92.60 ± 1.84	Bangladesh	97.59	78.61 ± 0.43	**82.45 ± 2.54**
Belarus	99.96	91.08 ± 3.08	**96.40 ± 1.76**	Paraguay	99.82	96.01 ± 1.23	**97.20 ± 0.35**
Serbia	99.76	**95.31 ± 0.48**	92.62 ± 3.83	Cyprus	90.07	**97.67 ± 0.34**	94.31 ± 7.21
Croatia	99.85	**94.59 ± 2.21**	88.04 ± 4.68	Bosnia	99.62	84.04 ± 1.47	**84.46 ± 5.76**
Greece	99.90	**81.98 ± 0.60**	40.97 ± 14.89	Ireland	96.52	**87.44 ± 0.69**	86.49 ± 1.31
New Zealand	99.73	94.27 ± 1.50	**99.44 ± 0.29**	Algeria	98.86	**85.37 ± 2.05**	84.65 ± 4.47
Portugal	99.59	**93.65 ± 0.46**	92.68 ± 1.46	Colombia	97.70	87.81 ± 0.92	**89.51 ± 0.88**
Bulgaria	99.24	91.03 ± 2.07	**93.47 ± 3.07**	Uzbekistan	99.80	**86.76 ± 1.13**	75.18 ± 1.92
Lithuania	99.99	**87.67 ± 3.05**	76.41 ± 1.66				

Tableau C.6 – Zero-shot transfer models' z-test significance test per country(**bold** value are rejected null hypothesis with $\alpha = 0.001$)

Country	Libpostal FastText	Libpostal BPEmb	FastText BPEmb	Country	Libpostal FastText	Libpostal BPEmb	FastText BPEmb
Belgium	88.71	92.22	4.70	Réunion	-10.61	-5.72	5.20
Sweden	80.64	55.54	-33.76	Moldova	15.11	17.78	3.56
Argentina	59.55	54.68	-6.27	Indonesia	29.42	26.00	-4.28
India	86.04	62.05	-28.71	Bermuda	10.42	8.95	**-1.75**
Romania	31.88	40.10	11.04	Malaysia	19.05	6.29	-14.26
Slovakia	60.15	42.70	-23.21	South Africa	7.77	6.16	**-2.12**
Hungary	108.57	143.25	45.25	Latvia	7.98	18.85	13.33
Japan	92.06	100.78	10.49	Kazakhstan	12.52	7.96	-6.12
Venezuala	21.12	25.46	5.56	New Caledonia	**1.63**	**2.22**	**0.68**
Philippines	49.84	44.09	-7.57	Estonia	11.87	16.18	5.78
Slovenia	20.74	14.76	-7.93	Singapore	11.19	10.97	**-0.29**
Ukraine	25.79	26.54	**1.03**	Bangladesh	12.36	10.65	**-2.05**
Belarus	26.44	16.41	-13.53	Paraguay	5.46	4.43	**-1.35**
Serbia	16.72	21.73	6.58	Cyprus	-6.47	**-3.23**	3.50
Croatia	17.03	26.36	12.39	Bosnia	10.49	10.33	**-0.22**
Greece	31.13	64.40	42.03	Ireland	5.98	6.43	**0.50**
New Zealand	15.49	**2.19**	-14.34	Algeria	8.68	8.95	**0.35**
Portugal	15.84	17.27	**1.85**	Colombia	6.44	5.65	**-0.91**
Lithuania	20.23	28.89	11.59	Uzbekistan	8.28	11.83	4.69
Faroe Islands	29.75	21.58	-10.92				

First, we observe that the **BPEmd** model reaches the highest accuracy most of the time. Indeed, 49% of the countries tested in zero-shot transfer reached a mean accuracy of at least

90% using **BPEmb** while using **fastText** only 46% of the countries reach that same accuracy. Most of these countries share either the same address structure or language proximity with training data. For instance, Venezuela shares the same address pattern as six other countries in the dataset and also shares the same language as Mexico, Spain, and the same Latin root as French. It was also interesting to observe that for Greece, **BPEmb** achieved near half the accuracy result as **fastText**. We hypothesize that for Greece fastText is able to produce better embeddings from subword units to reach this performance than **BPEmb**.

In contrast, the lowest results (below 70%) occur for countries where the address pattern and the country's official language were not seen in the training data such as India, Hungary, and Japan. The last two countries have had the lowest results of all. This is most likely due to the address structure (blue), which is the near inverse of the two most present ones (red and brown) (Figure C.2). Indeed, since most of the other patterns used for the training are the opposite, our models were not exposed to this pattern during training. Also, those two countries do not share language root with any of the ones present in the training data, which makes the task difficult for our models. We also see that Kazakhstan, which uses the same address pattern as Japan, achieves better results. The main difference is the official language (Kazakh and Russian) presence in the training dataset. Moreover, India achieves almost 20% better results than Hungary and Japan, even if Hindi does not occur in the training dataset. This is probably due to the use of a nearly identical address pattern as the first one (red). The only difference being the inversion of the province and the postal code. It could mean that if no shared language root is present, a shared address structure allows a decent parsing of the address (almost 70%).

Second, we observe that both models (**fastText** and **BPEmb**) obtain decent results compared to Libpostal, considering that none of these countries were seen during training. Also, an interesting result is that of the Reunion zero-shot accuracy. The fact that this country shares the same address pattern as France (which was part of the training set) explains the good performance which has surpassed that of Libpostal despite not being trained directly on data from Reunion as opposed to Libpostal.

Finally, we observe that **BPEmp** achieves better results than **fastText** for 21 out of 41 countries, where most of them are between 1% to 20% better. Considering that nearly 81% of the countries reach an accuracy above 80%, we conclude that using BPEmb embeddings gives good results for a zero-shot address parsing task considering that some languages and address patterns do not occur in the training data. To assess models' performance, we report the models' z-test significance test in Table C.6 using the same null hypothesis with $\alpha = 0.001$. We observe that **BPEmb** has a significantly better performance for 1 out of 41 countries compared to Libpostal, and we cannot reject the null hypothesis for 3 out of 41 countries, which means that performance between the two is not similar. Still, considering that the model was not trained, it's an interesting result. In addition, we observe that between **BPEmb** and

fastText we have 12 out of 41 countries where the performance is similar and 15 out of 41 are in the advantage of **fastText**. These results were surprising to us since **fastText** uses pre-trained embedding trained on a French corpus. We hypothesize that fastText can produce better generalization of embeddings using subword units than **BPEmb**. This highlights that our trained model to combine the **BPEmb** embeddings might have overfitted to our problem due to the dataset size in contrast with fastText embeddings.

C.7 Discussion

We estimate that we have reached our first objective, which was to build a model capable of learning to parse addresses of different formats and languages using a multinational dataset and subword embeddings. As for our attempt at zero-shot transfer learning, it yielded interesting results. These results give us insights into the direction that our future work should take. We would also like to state that due to the time and resources consuming nature of the training process, we have not been able to perform a grid search to find optimal hyperparameters for our models. It could be interesting to explore how other subword embeddings techniques, such as the character-based ones, would perform on the multinational address parsing task. Adding an attention mechanism (Bahdanau *et al.*, 2014) could also give interesting insights into the address elements on which the model focuses when making a tag prediction.

C.8 Conclusion

In this article, we tackled the multinational address parsing problem and proposed a solution based on the use of subword embeddings to solve the multilingual aspect of the problem, as well as a sequence-to-sequence model for the tagging task. Our approach was able to reach state-of-the-art results on address from twenty countries and despite the different address formatting systems without the use of any pre-processing nor post-processing. We have also explored the possibility of zero-shot transfer across countries and achieved interesting, but not yet optimal results. As part of our future work, we will aim at applying domain adaptation techniques to better transfer the learned knowledge about a country's address parsing to other countries' addresses.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada (IRCPJ 529529-17) and a Canadian insurance company. We wish to thank the reviewers for their comments regarding our work and methodology.

Bibliographie

"Classification and Regression Trees.

N. Abid, A. ul Hasan et F. Shafait : DeepParse : A Trainable Postal Address Parser. *In Digital Image Computing : Techniques and Applications)*, pages 1–8, 2018.

Elise Acheson, Michele Volpi et Ross S. Purves : Machine Learning for Cross-gazetteer Matching of Natural Features. *International Journal of Geographical Information Science*, pages 1–27, 2019.

Dzmitry Bahdanau, Kyunghyun Cho et Yoshua Bengio : Neural Machine Translation by Jointly Learning to Align and Translate, 2014.

Jean-Thomas Baillargeon : Utilisation de données textuelles dynamiques lors de la souscription de risques commerciaux. Mémoire de D.E.A., École d'actuariat de l'université Laval, Novembre 2018.

Gregory V. Bard : Spelling-Error Tolerant, Order-Independent Pass-Phrases via the Damerau-Levenshtein String-Edit Distance Metric. *In Proceedings of the Fifth Australasian Symposium on ACSW Frontiers*, volume 68 de *ACSW*, pages 117–124. Australian Computer Society, 2007.

Richard Bellman : Dynamic programming. *Science*, 153(3731):34–37, 1966.

Lasse Bergroth, Harri Hakonen et Timo Raita : A Survey of Longest Common Subsequence Algorithms. *In Proceedings Seventh International Symposium on String Processing and Information Retrieval.*, pages 39–48, September 2000.

Mikhail Bilenko et Raymond Mooney : Learning to Combine Trained Distance Metrics for Duplicate Detection in Databases. November 2002.

Piotr Bojanowski, Edouard Grave, Armand Joulin et Tomas Mikolov : Enriching Word Vectors with Subword Information, 2016.

Leo Breiman : Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

Eric Brill et Robert C. Moore : An improved error model for noisy channel spelling correction. *In Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics, October 2000.

Eric Busboom : US Address Parser, 2014. https://pypi.org/project/address-parser/.

Mary Elaine Califf et Raymond J. Mooney : Relational learning of pattern-match rules for information extraction. *In CoNLL97 : Computational Natural Language Learning*, 1997.

Patricio Cerda, Gaël Varoquaux et Balázs Kégl : Similarity Encoding for Learning with Dirty Categorical Variables, 2018.

Nicholas Michaelides Chair, Peter Brown, Francis Chacko, Mark Graham, Jeremy Haynes, David Hindley, Sheree Howard, Henry Johnson, Kathryn Pauly Morgan, Craig Pettengell, Richard Rodríguez et David Simmons : The Premium Rating of Commercial Risks. *In Proceedings of the General Insurance Convention*, pages 397–491, 1997.

Matt Chaput : Whoosh : Fast Pure-Python Full Text Indexing, Search and Spell Checking Library, 2017.

Peter Christen : Febrl - An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User Interface. pages 1065–1068, August 2008.

Peter Christen : Classification. *In Data Matching : Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, chapitre 6. Springer Science & Business Media, 2012a.

Peter Christen : *Data Matching : Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Science & Business Media, 2012b.

Peter Christen : Field and record comparison. *In Data Matching : Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*, chapitre 5. Springer Science & Business Media, 2012c.

Tim Churches, Peter Christen, Kim Lim et Justin Zhu : Preparation of Name and Address Data for Record Linkage Using Hidden Markov Models. *BMC medical informatics and decision making*, 2, January 2003.

William Cohen, Pradeep Ravikumar et Stephen Fienberg : A Comparison of String Metrics for Matching Names and Records. *Proceedings of the KDD Workshop on Data Cleaning and Object Consolidation*, October 2003a.

William W. Cohen, Pradeep Ravikumar et Stephen E. Fienberg : A Comparison of String Distance Metrics for Name-matching Tasks. *In Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB, pages 73–78. AAAI Press, 2003b.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer et Hervé Jégou : Word Translation Without Parallel Data, 2017.

Data Ladder : Delete Duplicated Software. https://dataladder.com/data-deduplication-software/delete-duplicates/, April 2020.

Marc-André Desrosiers : An Experience Rating Approach to Insurer Projected Loss Ratios. *In Actuarial Research Clearning House*, 2013.

DQ Global : Data Deduplication. https://www.dqglobal.com/solutions/business-needs/data-deduplication/, April 2020.

EGon : Data and Postal Address Deduplication Software. https://www.egon.com/solutions/deduplication, April 2020.

Mohammadreza Ektefa, Fatimah Sidi, Hamidah Ibrahim, Marzanah Jabar, Sara Memar et Abe Ramli : A Threshold-based Similarity Measure for Duplicate Detection. *In 2011 IEEE Conference on Open Systems, ICOS 2011*, pages 37–41, September 2011.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang et Chih-Jen Lin : LIBLINEAR : a Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, August 2008.

Jeremy Foxcroft, Adrian d'Alessandro et Luiza Antonie : Name2Vec : Personal Names Embeddings. *In Advances in Artificial Intelligence*, pages 505–510, April 2019.

Philip Gage : A New Algorithm for Data Compression. *C Users Journal*, 12(2):23–38, 1994. ISSN 0898-9788.

Ameya Godbole, Aman Dalmia et Sunil Kumar Sahu : Siamese Neural Networks with Random Forest for Detecting Duplicate Question Pairs, 2018.

Google APIs : Google maps platform. https://developers.google.com/maps/documentation/geocoding/, April 2020.

Yash Govind, Pradap Konda, Paul Suganthan G.C., Philip Martinkus, Palaniappan Nagarajan, Han Li, Aravind Soundararajan, Sidharth Mudgal, Jeff R. Ballard, Haojun Zhang, Adel Ardalan, Sanjib Das, Derek Paulsen, Amanpreet Singh Saini, Erik Paulson, Youngchoon Park, Marshall Carter, Mingju Sun, Glenn M. Fung et AnHai Doan : Entity Matching Meets Data Science : A Progress Report from the Magellan Project. *In Proceedings of the International Conference on Management of Data*, SIGMOD, pages 389–403. ACM, 2019.

Alex Graves, Santiago Fernández et Jürgen Schmidhuber : Bidirectional lstm networks for improved phoneme classification and recognition. *In* Włodzisław Duch, Janusz Kacprzyk, Erkki Oja et Sławomir Zadrożny, éditeurs : *Artificial Neural Networks : Formal Models and Their Applications*, pages 799–804. Springer Berlin Heidelberg, 2005.

Forest Gregg, Cathy Deng, Miroslav Batchkarov et Jean Cochrane : Parserator, 2014. https://parserator.datamade.us/.

Marti A. Hearst : Support Vector Machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998.

Benjamin Heinzerling et Michael Strube : BPEmb : Tokenization-free Pre-trained Subword Embeddings in 275 Languages, 2017.

Benjamin Heinzerling et Michael Strube : Sequence tagging with contextual and non-contextual subword representations : A multilingual evaluation. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, juillet 2019.

Daniel S Hirschberg : Algorithms for the Longest Common Subsequence Problem. *Journal of the ACM*, 24(4):664–675, 1977.

Sepp Hochreiter et Jürgen Schmidhuber : Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997a.

Sepp Hochreiter et Jürgen Schmidhuber : Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, novembre 1997b. ISSN 0899-7667.

Zainab Imtiaz, Muhammad Umer, Muhammad Ahmad, Saleem Ullah, Gyu S. Choi et Arif Mehmood : Duplicate Questions Pair Detection Using Siamese MaLSTM. *IEEE Access*, 8:21932–21942, 2020.

Insurance Bureau of Canada : *2019 Facts of the Property and Casualty Insurance Industry in Canada*. Ottawa, 2019.

Paul Jaccard : Distribution de la Flore Alpine dans le Bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–72, January 1901.

Matthew A. Jaro : Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou et Edouard Grave : Loss in Translation : Learning Bilingual Word Mapping with a Retrieval Criterion. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018.

Daniel Jurafsky et James H. Martin : *Language Modeling with N-Grams*, chapitre 3. Prentice-Hall, Inc., 2009a.

Daniel Jurafsky et James H. Martin : Logistic regressions. *In Speech and Language Processing (2nd Edition)*, chapitre 5. Prentice-Hall, Inc., 2009b.

Daniel Jurafsky et James H. Martin : Part-of-speech tagging. *In Speech and Language Processing (2nd Edition)*, chapitre 8. Prentice-Hall, Inc., 2009c.

Diederik P. Kingma et Jimmy Ba : Adam : A Method for Stochastic Optimization, 2014.

Xiang Li, Hakan Kardes, Xin Wang et Ang Sun : HMM-based Address Parsing : Efficiently Parsing Billions of Addresses on MapReduce. *In Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL, pages 433–436. ACM, 2014a.

Xiang Li, Hakan Kardes, Xin Wang et Ang Sun : HMM-Based Address Parsing : Efficiently Parsing Billions of Addresses on MapReduce. *In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 433–436. Association for Computing Machinery, 2014b. ISBN 9781450331319.

Larkin Liu, Yu-Chung Lin et Joshua Reid : Comparing the Performance of the LSTM and HMM Language Models via Structural Similarity, 2019.

Magellan Project : py_entitymatching. https://sites.google.com/site/anhaidgroup/projects/magellan/py_entitymatching, April 2020.

Bruno Martins : A supervised machine learning approach for duplicate detection over gazetteer records. *In GeoSpatial Semantics*, pages 34–51, 2011.

Tomas Mikolov, Kai Chen, Greg Corrado et Jeffrey Dean : Efficient Estimation of Word Representations in Vector Space, 2013a.

Tomas Mikolov, Kai Chen, Greg Corrado et Jeffrey Dean : Efficient Estimation of Word Representations in Vector Space, 2013b.

Shekoofeh Mokhtari, Ahmad Mahmoody, Dragomir Yankov et Ning Xie : Tagging Address Queries in Maps Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9547–9551.

Shekoofeh Mokhtari, Ahmad Mahmoody, Dragomir Yankov et Ning Xie : Tagging Address Queries in Maps Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9547–9551, July 2019.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute et Vijay Raghavendra : Deep Learning for Entity Matching : A Design Space Exploration. pages 19–34, May 2018.

Felix Naumann et Melanie Herschel : *An Introduction to Duplicate Detection*. Morgan and Claypool Publishers, 2010.

Openvenues : Pypostal a Python bindings to libpostal for fast international address parsing/normalization, 2016. https://github.com/openvenues/pypostal.

Frédérik Paradis : Poutyne : A Keras-like framework for PyTorch, 2018. https://poutyne.org.

Rebecca Passonneau, Nizar Habash et Owen Rambow : Inter-annotator agreement on a multilingual semantic annotation task. *In Proceedings of the Fifth International Conference on Language Resources and Evaluation*. European Language Resources Association, mai 2006.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai et Soumith Chintala : PyTorch : An Imperative Style, High-Performance Deep Learning Library. *In* H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox et R. Garnett, éditeurs : *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

Jeffrey Pennington, Richard Socher et Christopher Manning : Glove : Global vectors for word representation. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543. Association for Computational Linguistics, October 2014a.

Jeffrey Pennington, Richard Socher et Christopher Manning : GloVe : Global vectors for word representation. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, octobre 2014b.

John Pollard : *Premium Rating : Methods and Problems*, pages 174–193. Palgrave Macmillan UK, 1990.

Ronaldo Prati, Gustavo Batista et Maria-Carolina Monard : Data Mining with Imbalanced Class Distributions : Concepts and Methods. *In IICAI*, pages 359–376, 01 2009.

Philipp Probst et Anne-Laure Boulesteix : To Tune or Not to Tune the Number of Trees in Random Forest ?, 2017.

Laura Elena Raileanu et Kilian Stoffel : Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004.

Payam Refaeilzadeh, Lei Tang et Huan Liu : Cross-Validation. *Encyclopedia of Database Systems*, pages 532–538, January 2009.

Registraire des entreprises du Québec : *Banque de données publique sur les entreprises au Québec - Guide d'intégration fonctionnelle*. Québec, 2017.

Registraire des entreprises du Québec : Le registre et son contenu. http://www.registreentreprises.gouv.qc.ca/fr/a_propos/registre/, Avril 2020a.

Registraire des entreprises du Québec : Registre des entreprises du québec, 2020b. Données disponible en ligne sur le portail Données Québec, https://www.donneesquebec.ca/recherche/fr/dataset/registre-des-entreprises.

Graham Rhind : Global Sourcebook for International Data Management. https://www.grcdi.nl/gsb/global%20sourcebook.html, June 2020.

Eric Sven Ristad et Peter N. Yianilos : Learning String-Edit Distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(5):522–532, May 1998.

Hugo Robertson, Stephen Zaragoza : *The probabilistic relevance framework : BM25 and beyond*. Now Publishers Inc, 2009.

Stephen E. Robertson et Karen Sparck Jones : *Relevance Weighting of Search Terms*, page 143–160. Taylor Graham Publishing, 1988.

Frank Rosenblatt : The Perceptron : A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386, 1958.

Rasoul S. Safavian et David Landgrebe : A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):660–674, May 1991.

Claude Sammut et Geoffrey I. Webb, éditeurs. *TF–IDF*, pages 986–987. Springer US, 2010.

Mike Schuster et Kuldip K. Paliwal : Bidirectional Recurrent Neural Networks. *Transactions on Signal Proccessing*, 45(11):2673–2681, November 1997.

Vivek Sehgal, Lise Getoor et Peter D Viechnicki : Entity Resolution in Geospatial Data Integration. *In Proceedings of the 14th Annual ACM International Symposium on Advances in Geographic Information Systems*, page 83–90. Association for Computing Machinery, 2006.

George Gaylord Sempson : Holartic Mammalian Faunas and Continental Relationships During the Cenozoic. *GSA Bulletin*, 58(7):613–688, July 1947.

Rico Sennrich, Barry Haddow et Alexandra Birch : Neural machine translation of rare words with subword units. *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725, août 2016.

Valerie Sessions et Marco Valtorta : The Effects of Data Quality on Machine Learning Algorithms. pages 485–498, January 2006.

Shai Shalev-Shwartz et Shai Ben-David : Decision tree algorithm. *In Understanding machine learning : from theory to algorithms*, chapitre 18. 2014a.

Shai Shalev-Shwartz et Shai Ben-David : Introduction. *In Understanding Machine Learning : From Theory to Algorithms*, chapitre 1. Cambridge University Press, 2014b.

Shai Shalev-Shwartz et Shai Ben-David : Neural networks. *In Understanding machine learning : from theory to algorithms*, chapitre 20. 2014c.

S. Sharma, R. Ratti, I. Arora, A. Solanki et G. Bhatt : Automated Parsing of Geographical Addresses : A Multilayer Feedforward Neural Network Based Approach. *In IEEE 12th International Conference on Semantic Computing*, pages 123–130, 2018.

Shikhar Sharma, Ritesh Ratti, Ishaan Arora, Anshul Solanki et Gaurav Bhatt : Automated Parsing of Geographical Addresses : A Multilayer Feedforward Neural Network Based Approach. pages 123–130, January 2018.

Loveperteek Singh, Shreya Singh, Sagar Arora et Sumit Borar : One Embedding To Do Them All, 2019.

Amit Singhal : Modern Information Retrieval : A Brief Overview. *IEEE Data Engineering Bulletin*, 24:35–43, 2001.

Stephen Soderland : Learning Information Extraction Rules for Semi-Structured and Free Text. *Machine Learning*, 34(1–3):233–272, February 1999.

Statistique Canada : Tableau 33-10-0037-01 — nombre d'entreprises canadiennes, avec employés, Décembre 2017.

Giri Kumar Tayi et Donald P. Ballou : Examining Data Quality. *Commununications of the ACM*, 41(2):54–57, February 1998.

A. Viterbi : Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, septembre 1967. ISSN 0018-9448.

M. Wang, V. Haberland, A. Yeo, A. Martin, J. Howroyd et J. M. Bishop : A Probabilistic Address Parser Using Conditional Random Fields and Stochastic Regular Grammar. *In 16th International Conference on Data Mining Workshops*, pages 225–232, 2016.

Minlue Wang, Valeriia Haberland, Amos Yeo, Andrew Martin, John Howroyd et John Bishop : A Probabilistic Address Parser Using Conditional Random Fields and Stochastic Regular Grammar. December 2016.

Minlue Wang, Valerria Haberland, Amos Yeo, Andrew Martin, John Howroyd et Mark J. Bishop : A Probabilistic Address Parser Using Conditional Random Fields and Stochastic Regular Grammar. *In IEEE 16th International Conference on Data Mining Workshops*, pages 225–232, 2016.

Geoff Werner et Claudine Modlin : *Basic Ratemaking*, volume 4, chapitre 1. 2010.

R. J. Williams et D. Zipser : A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 1989.

William Winkler : String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods*, January 1990.

William E. Winkler : The State of Record Linkage and Current Research Problems. Rapport technique, Statistical Research Division, U.S. Bureau of the Census, 1999.

Ledell Wu, Adam Fisch, Sumit Chopra, Keith Adams, Antoine Bordes et Jason Weston : StarSpace : Embed All The Things ! 2017.

Sen Xu, Soren Flexner et Vitor R. Carvalho : Geocoding Billions of Addresses : Toward a Spatial Record Linkage System with Big Data. 2012.

Li Yujian et Liu Bo : A Normalized Levenshtein Distance Metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095, 2007.

Marijana Zekić-Sušac, Sanja Pfeifer et Nataša Šarlija : A comparison of machine learning methods in a high-dimensional classification problem. *Business Systems Research Journal*, 5(3):82 – 96, 2014.