Tolkienizer

LI, Xining MARTINEZ, Hermes MICKUS, Timothee

29 janvier 2018

1 Présentation générale

Le présent document correspond au rapport sur le développement et l'analyse du projet 2018 d'Industrialisation du TAL pour l'Université Paris Diderot. Le projet consistait en l'implémentation d'un script de pseudonymisation d'un sous-ensemble du corpus ENRON. Il reprend tous les éléments nécessaires à la compréhension de la tâche, du script implémenté, et à l'analyse des résultats produits.

1.1 Éléments introductifs

L'un des défis majeurs des applications de Traitement Automatique du Langage (TAL) est d'ordre éthique : comment garantir le droit à l'anonymat des particuliers lorsqu'on utilise comme matières premières leurs informations personnelles ? Cette question éthique est d'autant prégnante à ce jour que l'usage de données en quantités massives - on pensera évidemment aux applications "Big Data" - se répand et que le transfert d'informations s'accèlèrent - ce qu'a impliqué la révolution d'Internet. Ces nouvelles applications impliquent en outre le stockage et le traitement automatique des informations personnelles : si c'est une aubaine pour les domaines comme le TAL qui se consacrent à l'étude des informations que communique une personne, cela implique un débat sur les plans juridiques et éthiques quant à leur emploi à des fins qui peuvent être non seulement scientifiques, mais aussi commerciales, comme dans le cas du marketing ciblé.

Le problème de la confidentialité des données personnelles est ancien : pour le domaine médical, il remonte à la Grèce Antique et à Hippocrate. On comprend que des informations personnelles, confiées à un particulier dans un cadre précis, devraient d'un point de vue moral ne pas être divulguées. Cependant comment établir les limites de ce cadre ? Comment définir ce qu'est une information qui relève du domaine privé, et ce qu'est une information qui relève du domaine publique ?

L'anonymisation des données personnelles permet de résoudre une partie de ces problèmes : si l'on conserve les informations personnelles de chacun tout en empêchant l'identification desdites informations à la personne dont elles proviennent, on permet en un sens de protéger l'anonymat du particulier. En effet le particulier demeure certain que ses informations personnelles et privées ne sont pas connues comme étant les siennes. Si l'on divulgue les données personnelles, on empêche cependant qu'on puisse les associer à un individu particulier, ce qui rend moins saillant le problème éthique de cette divulgation.

En revanche, plusieurs difficultés peuvent poindre dès qu'on anonymise un texte, et notamment dans le cas des applications en TAL. On doit notamment pouvoir s'appuyer sur une définition claire de ce qui constitue une mention d'un particulier à anonymiser. Typiquement, toute information permettant l'identification doit être retirée : on pensera évidemment à supprimer toute information permettant l'identification de manière directe, comme les noms, prénoms, adresses physiques et électroniques, numéros de téléphones... Mais selon cette

logique, on devrait aussi retirer toute information permettant une identification indirecte : aussi les données qui, recoupées, permettent de distinguer un individu en particulier devraient de même être expurgées des données exploitées. Cependant ces données permettant une identification indirecte se retrouvent d'une part être difficles à détecter, d'autre part pontentiellement constituer l'intégralité des données exploitables. Une autre difficulté tout aussi évidente est que retirer toute mention d'un particulier rend la lecture difficile, et conduit aussi à l'introduction d'un artefact dans les données que traitera l'application.

S'il est difficile de rémédier à la première de ces deux problématiques, la seconde, en revanche, peut trouver une réponse dans la pseudonymisation. Plutôt que du supprimer brutalement les références directes, on préférera leur substituer des informations neutralisées, ne correspondant à aucun individu réel. On souligne aussi souvent, en outre, que la première problématique est difficile à exploiter, car utiliser de telles données pour une identification indirecte demande des moyens conséquents. La pseudonymisation s'avère donc, en pratique, une solution envisageable à la question éthique de la protection des informations privées.

1.2 Définitions

Nous utiliserons les termes suivants au cours de ce rapport :

Traitement automatique du Langage (TAL) Domaine scientifique et technologique se rapportant à l'étude des traces langagières de manière automatique (en particulier informatique).

Entités Nommées ("*Named Entities*", NE) Référence dans un texte à un individu du monde réel, ou permettent d'identifier un individu du monde réel particulier. En particulier : noms, prénoms, noms d'entreprises ou de lieux, adresses physiques ou éléctroniques (emails, URL et IP), numéros de téléphones ou identifiants administratifs.

Reconnaissance d'Entités Nommées ("*Named Entities Recognition*", NER) Tâche de TAL correspondant à indiquer, dans un texte donné, quelles NE sont présentes, et à quel endroit du texte.

Anonymisation Tâche de TAL correspondant à la suppression des Entités Nommées dans un texte.

Pseudonymisation Tâche de TAL correspondant au remplacement des Entités Nommées dans un texte par une séquence ne référant à aucun individu du monde réel.

Coréférence Les NE qui renvoient au même individu réel partagent la même coréférence.

Forme de mention Les chaînes de caractères de NE qui coréfèrent au même individu (en particulier, à la même personne) sont des formes de mentions du même individu.

1.3 Présentation du projet

1.3.1 Présentation de la tâche spécifique

Ce projet consiste en l'implémentation d'un outil pour pseudonymiser un sous-ensemble du corpus ENRON, sans rien perdre des informations pragmatiques quant aux différents locuteurs (locuteur, allocutaire, coréférences, registres de langue, anaphores...). Ce corpus correspond à une collection d'emails envoyés et stockés par la société Enron et sortis du domaine privé. La tâche inclut également une évaluation des performances de l'outil, les mo<yens qu'il met en oeuvre et une analyse de ses limites.

1.3.2 Données à traiter

Le corpus ENRON est une collection de mails écrits et reçus par la société Enron. Il a été constitué au cours de la banqueroute de la société due à des manipulations fiscales frauduleuses, afin d'alimenter l'enquête subséquente au dépôt de bilan. Tombé dans le domaine public suite à son usage juridique, la taille importante du corpus (160 Go, soit plus de 600,000 emails) l'a rendu assez difficile à manipuler. L'extrait du corpus à traiter ici ne contient que 7,875 emails. Nous avons recensé dans ceux-ci 31,148 mentions d'organisations (3,845 mentions distinctes), 41,185 mentions de lieux (2,734 mentions distinctes) et 121,105 mentions de personnes (8,549 mentions distinctes).

L'intérêt d'un tel corpus est qu'il est constitué de données réelles, présentant ainsi des contextes variés d'occurences de NE, ainsi que des défis importants en terme de résolutions de coréférences de ces NE. Le corpus ENRON est par ailleurs l'un des rares corpus massifs de mails à l'origine privés accessibles publiquement.

2 Implémentation

2.1 Choix des outils

Une série de choix techniques a guidé l'implémentation de ce programme.

L'un des premiers sujets de questionnement que nous avons eu a été celui du langage de programmation. Nous nous étions déjà préalablement familiarisé avec Java et Python (et notamment sa librairie NLTK). Nous avons finalement choisi d'utiliser le langage de programmation Groovy, pour plusieurs raisons. Premièrement, la syntaxe était assez permissive et similaire à Java pour ne pas poser de défi technique. Secondement, Grapes, qui est intégré dans la distributuion standard de Groovy, permet une gestion de projet et des dépendances très simple. Troisièmement, l'interfaçage avec les librairies Java se fait sans aucune

difficulté en Groovy. Enfin, l'existence d'un interpréteur pour Groovy permet une prise en main rapide.

Une seconde question que nous avons abordée concernait le choix des librairies externes. Nous en avons comparé deux en particulier : Apache OpenNLP et Stanford CoreNLP. Elles nous semblaient toutes deux pertinentes dans le cadre de ce projet, car proposaient toutes deux des fonctionnalités pour la tâche NER. Nous avons conclu qu'en termes de performances et de maniabilités, la librairie Stanford CoreNLP était supérieure à Apache OpenNLP. Plus précisément, Apache OpenNLP ne proposait de détection que pour les seules personnes, tandis que Stanford CoreNLP permettait la détection de lieux, personnes et organisations à l'aide d'un premier module, et emails et URL à l'aide d'un second. Les librairies Python que nous avons étudiées étaient moins performantes en termes de temps de calcul, ce qui nous a conforté dans notre choix d'utiliser Groovy.

La dernière question technique que nous mentionnons ici a été de savoir par quoi remplacer les NE que nous aurions extraites. La suppression pure et simple de celles-ci correspondrait à une tâche d'anonymisation. La subtstition par un label par défaut pour chaque élément aurait été envisageable, mais nuisait beaucoup à l'idiomaticité du corpus. Notre choix s'est donc porté sur un remplacement des références à des individus réels par des références à des individus fictifs. Il nous fallait donc un univers fictif qui remplissent plusieurs critères : il devait à la fois être très fourni en noms et en lieux, ainsi qu'être suffisamment répandu et accessible pour que nous puissions trouver des ressources libres pour ce projet. L'univers fictif qui répondait le mieux à ces critères était celui de J.R.R. Tolkien ; en effet le volume de ses oeuvres se chiffre en milliers de pages (sans même prendre en compte ses notes), et il existe une importante communauté de fans et de nombreux sites internets qui y sont dédiés.

Enfin, nous avons mis en place un dépôt git afin de disposer d'un système de versionnage pour ce travail en commun. Il est accessible à cette URL.

2.2 Constitution des ressources nécessaires

Les différents choix techniques que nous avons faits nous ont conduit à créer manuellement plusieurs ressources.

Le plus urgent a été la collection de noms de lieux et de personnes issues de l'univers de J.R.R. Tolkien. Pour ce faire, nous avons capturé la structure de pages issues de ressources libres. Les noms de personnes ont étés récupérés du DOM la page Behind The Name ; les noms de lieux de la structure wiki de la page Tolkien Gateway et du dom de plusieurs pages issues du site The encyclopedia of Arda. Les DOM ont ensuite été transformés à l'aide d'utilitaires UNIX, notamment grep et sed. Nous avons converti la structure tabulaire du tableau de Behind The Name en format csv, utilisant la tabulation comme séparateur. Certains noms de familles (notamment les indices dynastiques comme "Thorin III") ont été supprimés pour des raisons d'utilité à la tâche. Ces informations ont été sauvegardées dans le fichier names.txt. Les noms de lieux de Tolkien Gateway ont été manuellement normalisés, puis triés et rendus uniques à l'aide

des commandes UNIX sort et uniq. La liste en sortie a été sauvegardée dans le fichier places.txt. Les deux fichiers ont été concaténés pour produire le fichier voc.txt, que nous utilisons pour la production de noms d'organisations.

Le second type de ressources que nous avons traité consiste en l'extraction de modèles et la compilation d'expressions régulières pour la librairie Stanford CoreNLP. Ces éléments sont étudiés par les classifieurs isssus de la librairie : les modèles permettent l'emploi direct, sans entraînement, du classifieur CRF (préfixé ll dans notre code) ; tandis que les expressions régulières servent pour le classifieur basé sur des expressions régulières (préfixé rl dans notre code).

2.3 Architecture

Le programme en lui-même se constitue de deux fichiers, extractor.gy et sudonizing.gy. Le second gère la pseudonymisation des NE extraites. Le premier permet de gérer le processus itératif : lire les mails source, extraire les NE à pseudonymiser, gérer l'appel au second fichier pour pseudonymiser les NE, puis enregistrer les mails pseudonymisés. Les deux fichiers groovy sont déclarés dans le même package afin d'en simplifier l'usage. Les ressources utilisées par les scripts sont consignés dans le répertoire ressource.

Le script extractor.gy gère le déroulément essentiel de ce projet : un script simple et précis sur plusieurs étapes pour achever la pseudonymisation du corpus Eron. Il contient en outre du déroulement itératif pluisieurs fonctions supplémentaires pour parcourir listes et hashmaps. Le déroulement itératif commence avec une fonction loadMailsToArray qui charge les fichiers sources dans une liste, et garde les path de chaque email ansi que l'ordre des dossiers parcourus. Secondement, on génère un classifieur CRF à l'aide du framework coreNLP capable de repérer trois types de NE (Personne, Lieu, Organisation) à l'aide d'une fonction createNElist. On utilise aussi des properties (liste de modèles de coreNLP) pour associer un genre à chaque NE de type Personne trouvée. Pour les NE restantes (de type email) la fonction createSuplementaryNEList crée a partir d'un fichier properties de regex une liste de NE. On passe alors a la génération de pseudonymes tirés de la literature de tolkien avec l'aide des classes déclarées danssudonizing.gy (cf. infra) : la fonction createTolkienList prend en argument la liste des NE précedemment extraites et génère une liste équivalente "tolkienisée". Celle-ci est mise en regard de la liste de NE originelle pour enfin créer une nouvelle liste de mails pseudonymisés via la fonction tolkienize qu'on écrit dans un fichier en sortie.

Le fichier sudonyzing.gy contient plusieurs classes pour gérer la pseudonymisation des NE. La classe PersonWrapper est une classe cannonique qui permet simplement de simplifier la lecture pour les NE référent à des êtres humains.

Les autres classes implémentent toutes l'interface Sudonizer, qui typologiquement définit un objet qui peut substituer de manière systématique un pseudonyme à une NE. Cette interface déclare deux méthodes : sudonize(rawString), qui correspond à pseudonymiser un exemple, et getRegister(), qui correspond à obtenir le registre de correspondances entre NE et pseudonymes.

La classe SimpleSudonizer est l'implémentation la plus basique de cette interface. Elle correpsond à la substition systématique et directe d'une chaîne de caractère par une autre, et ne met en jeu aucun autre mécanisme.

La classe AwareSudonizer est une implémentation qui permet de pseudonymiser en étant conscient des pseudonymes précédemment donnés. Elle est conçue comme un wrapper : elle se base sur la prédiction d'un autre objet Sudonizer si elle ne trouve aucun pseudonyme dans le registre. L'intérêt de cette classe est par ailleurs que le registre peut être global, ce qui permet alors de proposer un pseudonyme qui soit consistent avec l'ensemble des pseudonymes des Sudonizer partageant le même registre global.

La classe PersonSudonizer est une implémentation de l'interface Sudonizer spécifiquement dédiée à la pseudonymisation des êtres humains. Elle permet de conserver les informations quant au genre de la personne, ainsi que de conserver la forme de la mention, et ce de manière systématique. Différents individus partageant le même prénom partageront (au moins en partie) le même pseudonyme, et différents individus possédant le même nom de famille partageront le même nom de famille en pseudonyme. Qui plus est la forme de la mention est respectée : si l'on mentionne seulement un prénom ou un nom, alors le pseudonyme associé n'est composé que d'un prénom ou d'un nom. Parreillement si l'on mentionne une identité complète, alors le pseudonyme retourné contiendra un prénom et un nom.

La classe OrgSudonizer est une implémentation qui permet de pseudonymiser un nom d'organisation. Elle repose sur un mécanisme génératif : les pseudonymes qu'elle propose sont construits semi-aléatoirement, en associant deux noms tirés d'un fichier vocabulaire et un suffixe tiré d'une brève liste pour rendre lisible le fait qu'il s'agisse d'un nom d'entreprise. En effet nous n'avons pas trouvé dans l'univers de J.R.R. Tolkien un ensemble d'organisations (qu'il s'agisse d'ethnies, de pays, de magasins ou autres) qui fut assez grand pour s'appliquer à l'ensemble des noms d'organisations présent dans notre corpus. La stratégie générative s'est donc avérée ici nécessaire.

2.4 Emploi du script

Le script s'emploie simplement en appelant le script principal : groovy extractor.gy, pour peu qu'on ait déjà compilé les sources : groovyc extractor.gy sudonizing.gy. Le seul prérequis est d'avoir groovy installé.

3 Discussion

3.1 Limitations du script implémenté

Le script possède plusieurs limitations dues à sa conception d'une part, dues aux outils d'autre part, et enfin, dues à la tâche de pseudonymisation elle-même.

Les limitations internes à l'outil peuvent se distinguer en deux groupes. Premièrement, le script est basé sur la lecture de ressources externes En particulier

les classes pour la pseudonymisation (et en particulier SimpleSudonizer) sont extrêmement dépendantes d'un vocabulaire préalablement constitué, et ce pour toutes les substitutions effectuées. Un exemple parlant est celui des noms de lieux. En effet, nous avons recensé 2,734 noms de lieux dans le corpus originel ; mais notre base de données n'en comporte que 1,834. Concrètement, cela signifie que des noms de lieux différents dans le corpus originel peuvent (et vont) recevoir le même pseudonyme. L'un des moyens de contournement que nous aurions pu mettre en place aurait été un modèle génératif, comme nous l'avons fait pour les noms d'organisations. Un détail du vocabulaire nous a cependant arrêté : les noms listés peuvent contenir des indications topographiques (comme "River", "River of", "Mountain of"...) ; aussi une stratégie générative ne nous garantie ni cohérence ni lisibilité des pseudonymes.

La seconde catégorie de limitations internes au script est due aux besoins spécifiques à un corpus. Si l'on regarde les différentes classes pour la pseudonymisation, on comprend qu'elles sont avant tout définies pour leur usage. De celà, il suit que le script ne peut traiter que les types d'entités qui ont été définies pour le corpus. En d'autres termes, le script est spécifique à ce sous-ensemble du corpus ENRON, et à l'analyse que nous en avons fait. Si certaines de ces analyses peuvent être modifiées aisément grâce au système de properties Regexp, d'autres sont intrinsèquement liées au code lui-même.

Un exemple possible est celui des noms : bien qu'il y ait une forme de génération dans la composition de l'identité, les différentes formes de mentions sont dépendantes de ce que l'on possède comme information. Les initiales par exemple ne sont pas gérées, ou alors en tant que prénom à part entière, auquel cas on perd le lien entre les formes de mention "S. Scott" et "Susan Scott". De même les deuxièmes prénoms sont purement ignorés, puisque les formes de mentions sont trop peu régulières pour que nous puissions distinguer un nom d'un prénom. En bref, les pseudonymes proposés par le script évoquent en partie seulement les NE extraites, et ce suivant des décisions préalables à l'implémentation.

Les limitations inhérentes à la qualité et la forme de la sortie des outils forment un second ensemble de difficultés. Les classifieurs de Stanford CoreNLP fonctionnent selon des algorithmes *conditionnal random field* ou *conditional markov model*. Par conséquent les résultats sont d'un ordre statistique, et non déterministe, conduisant parfois à des faux négatifs (NE non détectées), des faux positifs (NE détectée à tort), ou des mauvaises classifications (type de NE mal détecté). Enfin les prédictions se faisant au niveau du token, les limites d'une forme de mention de NE sont parfois dures à établir dans le cas où plusieurs NE du même type se suivent.

Dernièrement, il existe des limitations dûes à la tâche de pseudonymisation elle-même. Il est difficile d'établir exactement toutes les formes que peuvent prendre des informations privées permettant l'identification directe dans un texte libre. Les documents suivants un format strict (formulaire, etc.) peuvent être assez trivialement pseudonymisés, cependant des productions libres sont toujours susceptibles de donner des informations permettant l'identification directe d'une manière inattendu. L'exemple le plus évident est la description péri-

phrastique, qu'il est impossible de détecter via une tâche de NER ; Un autre cas plus fréquent encore, et plus pertinent dans le cadre de la pseudonymisation, est celui du surnom, qui peut avoir comme base un nom autant qu'une description univoque de l'individu en question. Aussi la tâche de pseudonymisation d'un texte libre est vouée à se limiter à ce que l'on peut formellement désigner.

3.2 Résultats obtenus

Nous n'avons pas rencontré de problèmes d'encodages spécifiques, ce qui nous a permis de traiter l'entièreté du corpus. Nous avons traité les types de NE suivants : Personne, Lieu, Organisation et e-mail. Par conséquent tout ce qui ne correspondait pas à l'une de ces catégories n'a pas été pseudonymisée. En particulier les numéros de téléphones, ou identifiants administratifs ou autre n'ont pas été traités par ce script.

Nous avons étudié une centaine de mails pseudonymisés pour nous faire une idée de la validité des résultats obtenus. Pour les NE que nous extrayons, nous avons eu une vingtaine de cas *stricto sensu* mal détectés. Les problèmes associés sont en général des mauvaises délimitations des NE qui inclut du texte à ne pas pseudonymiser, des mauvaises classifications dues à un token ambigu, ou plus rarement des mauvaises délimitations par tokens manquants, en général lié à la présence d'un token ambigu. Les cas de NE entièrement non détectés sont plus fréquents, nous comptons au total aux alentours de 89 % de précision. Ils peuvent correspondent à plusieurs schémas : soit des noms isolés (signatures, apostrophes en début de mails, etc.) soit des surnoms mal identifiés (souvent à cause de ce qu'ils sont orthographiés en minuscules), soit des cas ambigus. Le classifieur par expression régulière implémenté dans le script extractor.gy aurait normalement pu détecter ces cas de manière globale, pour peu qu'on ajoutât manuellement les expressions régulières aux ressources. Le temps nous a cependant manqué pour effectuer cette maintenance nécessaire.

La pseudonymisation des NE extraites n'a pas posé de problèmes, du moins dans le cadre défini par les limitations exposées ci-dessus. Les noms de lieux détectés, en particulier et comme déjà mentionnés plus haut, ne sont pas systématiquement associés à des pseudonymes distincts. Le même problème peut théoriquement se poser pour les NE de type Personne, mais le problème n'a pas été avéré dans l'étude rapide du corpus que nous avons faite. Notons que pour qu'un cas de deux NE Personnes associées au même pseudonyme se présente, il faudrait que les nombres de NE Personnes déjà détectées respectifs à chacune soient congruents modulo le nombre de noms de familles et modulo le nombre de prénoms, ce qui rend le cas relativement peu probable. Un élément que nous pouvons cependant souligner à ce sujet est que nous n'avons pas su faire correspondre de très près les emails et les personnes, et ce malgré l'emploi d'un registre global.

4 Conclusion

Le présent rapport a pu mettre en avant deux points majeurs.

D'une part, qu'il est d'une part difficile de définir strictement et formellement tout ce qui doit être pseudonymisé dans un corpus, ni dans quelle mesure le lien entre différentes formes de citations doit être conservé. Rajouter des contraintes pour pouvoir gérer tel ou tel cas de figure conduit souvent à des modèles de plus en plus complexes ; et pouvoir faire le lien entre entités de types différents de manière cohérente s'avère rapidement peu gérable. En dehors de la question de l'implémentation, les surnoms, abbréviations, libertés orthographiques ou emplois inattendus (on pensera par exemple aux noms de répertoires qui peuvent être nommés d'après une personne, une entreprise...) introduisent un degré de variation qu'il faut aussi prendre en compte. Enfin la question de quelle information il est envisageable de perdre n'est pas non plus annodine : les différentes formes de mentions d'une NE doivent(elles être systématiquement distinguées ? faut-il conserver et pseudonymiser les métadonnées ?

D'autre part, on touche aussi ici aux limitations des outils de NER. Nous avons décidé d'employer deux classifieurs, lesquels se sont avérés, après usage, posséder des défauts assez peu contournables. Le classifieur CRF est d'ordre statistique, et non pas déterministe. Dans le cadre de données sous format libre, en particulier, dès lors qu'on atteint un volume conséquent, on obtient la certitude de faux négatifs : des NE qu'on devrait pseudonymiser, et qui ne le sont pas. Si l'exigence est d'ordre moral, éthique ou juridique comme c'est souvent le cas dans les tâches d'anonymisation, cette caractéristique est très peu appréciable. Le classifieur par expression régulière évite cet écueil, en revanche il réclame une mise en place très coûteuse. Pour être précis, le classifieur doit posséder pour chaque élément qui doit être pseudonymisé une règle qui y correspond, et ces règles sont pour l'essentiel écrites et intégrées manuellement. Conjugué au point précédent que nous soulevions : la multiplicité des formes de mentions et la difficulté de la formalisation de celles-ci, on comprend alors que la somme de travail à investir croît presque exponentiellement avec chaque nouveau cas de figure.

En résumé, le travail présenté ici présente les points problématiques saillants d'une tâche de pseudonymisation ; et propose de plus plusieurs outils pour les résoudre.