Institut National des Langues et Civilisations Orientales

Département Textes, Informatique, Multilinguisme

Étude sur l'apport de la sélection des caractéristiques dans la classification multi-classe des textes

MASTER

TRAITEMENT AUTOMATIQUE DES LANGUES

Parcours :

Ingénierie Multilingue

par

Yuming ZHAI

Directeur de mémoire :

Damien Nouvel

Encadrant :

Gaël Patin

Année universitaire 2015/2016

Remerciements

Dans un premier temps, je tiens à remercier sincèrement toute l'équipe pédagogique de Traitement Automatique des Langues du laboratoire ER-TIM, pour leur formation de haute qualité, leur patience et leur aide permanente tout au long du master.

J'exprime toute ma reconnaissance à mon directeur de mémoire, Damien Nouvel, qui m'a donné des conseils et des remarques pertinents et m'a permis de clarifier mes idées. Je le remercie également pour sa disponibilité et son attention aux détails.

Je remercie tout particulièrement mon encadrant de stage, Gaël Patin, qui m'a patiemment guidée et encadrée sur différentes tâches durant ces trois stages consécutifs chez XiKO. Ses connaissances étendues, son humour, sa rigueur et sa méthodologie de travail m'ont aidée à mener à bien ce travail.

J'adresse mes profonds remerciements à tous mes camarades et mes collègues de travail, pour leur aide et l'ambiance agréable qu'ils apportent tous les jours.

RÉSUMÉ

Ce travail porte sur l'étude de l'apport de la sélection de caractéristiques pour la classification multi-classe de textes. Quatre méthodes de sélection ont été comparées : la spécificité lexicale, le TF-IDF, l'information mutuelle, et la différence proportionnelle catégorique. Pour éviter l'effet du sur-apprentissage, la sélection des caractéristiques a été intégrée à la validation croisée pour chaque sous-échantillon d'apprentissage. L'évaluation est réalisée principalement par un algorithme Bayésien Naïf Multinomial, et des tests sur les Machines à Vecteurs de Support ont été menés pour étudier l'effet de la régularisation. Selon des résultats expérimentaux, la spécificité lexicale, qui a obtenu une micro-moyenne F-mesure de 72.14% avec une réduction de 57% de caractéristiques, est la méthode la plus performante.

Mots-clés : classification multi-classe des textes, sélection de caractéristiques, apprentissage supervisé, validation croisée, régularisation

TABLE DES MATIÈRES

Liste des figures

Liste des tableaux

INTRODUCTION

Ce travail a été réalisé au sein de l'entreprise XiKO [1], une entreprise spécialisée dans l'étude des Big Data conversationnels sur le web, dont le service KOVERI^{TM} propose une solution de monétisation des contenus sur les pages web pour le ciblage publicitaire. Le but est de livrer des données analysées sémantiquement, divisées en segments contextuels et comportementaux. Les segments contextuels désignent des thématiques mentionnées dans des pages web, et les segments comportementaux portent sur la sémantique des contenus lus pendant la navigation, plus précisément composés de l'intérêt de l'internaute et de l'intention d'achat. Par ces moyens on cherche à augmenter la capacité de ciblage pour les annonceurs, et à améliorer la monétisation des emplacements sur les pages web pour les éditeurs. Dans ce contexte, l'objectif de ce présent travail est d'améliorer la performance du classifieur multi-classe interne pour attribuer aussi précisément que possible des segments contextuels aux pages web.

La classification des textes est une tâche qui vise à classifier les nouveaux documents dans des catégories prédéfinies en s'appuyant sur le contenu ou sur des métadonnées. Étant donné le volume toujours plus important des données textuelles non structurées en ligne, cette technique est devenue essentielle pour organiser et traiter ces données. Dans notre cas d'étude, pour une page web, on cherche à attribuer une classe parmi les 27 catégories de premier niveau dans notre taxonomie des segments de marketing, ce qui revient à élaborer une classification multi-classe où les classes s'excluent.

Parmi les travaux réalisés dans ce domaine, plusieurs algorithmes d'apprentissage automatique ont été utilisés pour entraîner des classifieurs sur des exemples annotés, à savoir la méthode des k plus proches voisins (k-NN) [Yang and Pedersen, 1997], le réseau de neurones [Wiener and Weigend, 1995], les machines à vecteurs de support [Joachims, 1998, Drucker et al., 1999, Dumais et al., 1998, Klinkenberg and Joachims, 2000], la régression logistique [Ifrim et al., 2008], l'algorithme bayésien naïf [Larkey and Croft, 1996], les arbres de décision [Lewis and Ringuette, 1994], etc.

Une caractéristique ou difficulté majeure de la classification des textes sur la base de l'apprentissage est la grande dimensionnalité de l'espace des caractéristiques, qui peut contenir des dizaines ou des centaines de milliers de mots distincts figurant dans le corpus et s'avérer beaucoup trop nombreux pour certains algorithmes d'apprentissage [Yang and Pedersen, 1997]. Dans le même temps, l'exactitude pourrait être diminuée si des caractéristiques qui généralisent mal ou qui sont bruitées étaient inclues dans le modèle.

Ce que l'on cherche à étudier à travers ce travail, c'est l'apport de la sélection de caractéristiques pour améliorer la rapidité de la classification, et dans quelle proportion des caractéristiques peuvent être exclues sans dégrader le résultat. Le principe de la sélection est que des méthodes prédéfinies vont attribuer à chaque caractéris-

1. http://www.xiko.fr/

tique un score, et celles considérées comme les plus pertinentes vont être extraites dans un sous-ensemble pour construire un modèle de prédiction. Après la sélection des caractéristiques, la performance en termes de temps et/ou d'espace sera améliorée si moins de calculs et/ou de mémoire sont requis pour catégoriser un corpus [Forman, 2007].

Les hypothèses sur les avantages de la sélection de caractéristiques sont multiples : la réduction du nombre de caractéristiques peut pallier le problème du sur-apprentissage afin d'obtenir des modèles plus généralisés ; la diminution du temps d'exécution ; et une meilleure lisibilité et interprétation sur les caractéristiques choisies selon les classes. On se pose aussi, dans les cas où on n'effectuerait qu'une sélection basique sans aucune méthode prédéfinie, la question suivante : quels seraient les résultats par le Bayésien Naïf Multinomial et les SVM qui disposent des paramètres de régularisation pour limiter le sur-apprentissage ?

Ce document est organisé comme suit : dans le premier chapitre nous présentons les travaux précédents et notre positionnement. Dans le chapitre 2, nous présentons les méthodes de sélection implémentées dans ce travail. L'algorithme du Bayésien Naïf Multinomial et des Machines à Vecteurs de Support sont présentés dans le chapitre 3. Le corpus, les paramètres du classifieur et l'algorithme sont décrits dans le chapitre 4. Nous présentons nos expériences et les résultats obtenus dans le chapitre 5, suivi par des discussions dans le chapitre 6, et nous concluons dans le chapitre 7.

CHAPITRE 1

ÉTAT DE L'ART

Les avantages d'un classifieur construit par apprentissage automatique par rapport à l'approche de l'ingénierie de connaissances (qui consiste en la définition manuelle d'un classifieur par des experts du domaine), repose sur leur plus grande efficacité et leur directe portabilité dans d'autres domaines ; ainsi on peut faire l'économie d'un travail « à la main ». Une présentation des différentes recherches dans ce domaine se trouve dans l'ouvrage « *Machine Learning in Automated Text Categorization* » de Sebastiani [Sebastiani, 2002].

Pour diminuer la variance du classifieur, c'est-à-dire pour obtenir les mêmes décisions sur des données d'apprentissage légèrement différentes, on peut choisir la régularisation ou la sélection de caractéristiques. Face aux caractéristiques qui peuvent faire l'objet d'un sur-ajustement, il s'agit soit de diminuer leur poids, soit de les supprimer directement [Jurafsky and Martin, 2015, chapitre 7]. Dans notre contexte de travail, le classifieur principal est le Bayésien Naïf Multinomial qui ne dispose pas de paramètre de régularisation ; en conséquence, la sélection de caractéristiques est utilisée pour ne conserver que celles considérées comme informatives.

La sélection de caractéristiques impose l'hypothèse qu'il existe des caractéristiques non pertinentes. Cependant Joachims [Joachims, 1998] montre que même s'il n'utilise que des caractéristiques avec des rangs bas, classées par le score de gain d'information, le résultat est meilleur que celui obtenu par une classification aléatoire. Joachims compare les algorithmes d'apprentissage automatique tels que les SVM, le Bayésien Naïf, le Rocchio, le k-NN et l'arbre de décision C4.5 en sélectionnant les caractéristiques par la méthode du gain d'information sur le corpus Reuters-21578 (135 classes liées à l'économie) [Reuters-21578, 1997] et OHSUMED (titre ou titre et résumé des journaux médicaux, 14 321 classes) [Ohsumed, 2005]. Les SVM avec le kernel RBF s'avèrent le meilleur algorithme. De plus sa bonne performance est constante et robuste, avec une bonne généralisation dans une grande dimensionnalité ; cela peut éliminer le besoin de la sélection de caractéristiques. La conclusion de Joachims est qu'un bon classifier doit combiner beaucoup de caractéristiques et qu'une sélection agressive conduira dans la grande majorité des cas à une perte d'information.

Les expériences de Yang et Pedersen [Yang and Pedersen, 1997] ont pour but de tester la performance des classifieurs après une réduction agressive de la dimensionnalité de l'espace des caractéristiques. Ils se penchent sur un problème de classification multilabel [1], menée sur les corpus Reuters-22173 (92 catégories) et OHSUMED. Le résultat est une liste ordonnée de catégories, chacune associée avec un score de confiance. Les deux algorithmes utilisés sont k-NN et l'ajustement par les moindres

1. classification multilabel : possibilité de choisir zéro ou plusieurs étiquettes par objet à classifier

carrés linéaires (LLSF), avec dans les deux cas une prise en compte du contexte. Les auteurs indiquent aussi que ces deux algorithmes ne supposent pas d'indépendance entre les termes ni entre les classes. Cinq méthodes de sélection sont comparées : la fréquence du document (DF), le gain d'information (IG), l'information mutuelle (MI), le test chi-deux (CHI) et la force des termes (TS). Leurs expériences montrent que l'IG et le CHI ont été les plus efficaces avec une réduction agressive des termes sans perdre d'exactitude de classification.

Afin d'obtenir une évaluation globale de la performance à l'échelle du document, Yang et Pedersen ont choisi les caractéristiques globalement. C'est-à-dire que pour chaque caractéristique, ils ont gardé seulement son score maximal en combinaison avec une certaine catégorie ; en même temps, ils ont aussi combiné les scores et calculé leur moyenne selon la distribution des catégories. Ceci n'est pas le cas de notre étude dans laquelle nous avons choisi les caractéristiques spécifiques à chaque classe, où chaque caractéristique reçoit un score de mesure par rapport à chaque catégorie.

De fortes corrélations ont été trouvées entre l'IG, le DF et le CHI, qui peuvent tous éliminer au moins 90% ou plus de termes uniques avec une performance meilleure ou égale que celle avec l'utilisation de toutes les caractéristiques. Les auteurs ont montré que cette corrélation n'était pas dépendante de leur corpus. Contrairement à la recherche d'information, les termes communs s'avèrent informatifs pour la catégorisation des textes, mais évidemment cette affirmation ne prend pas en compte les mots-outils. L'information mutuelle (MI) obtient de faibles performances à cause de son biais de préférence pour les termes rares et sa sensibilité aux erreurs d'estimation de probabilité [Yang and Pedersen, 1997].

Forman [Forman, 2003] compare douze méthodes de sélection sur dix-neuf jeux de données de classification multi-classe, parmi lesquelles la méthode BNS (*Bi-Normal Separation*) qu'ils proposent dépasse les autres dans la majorité des cas avec une marge substantielle, surtout quand il existe un grand déséquilibre dans la distribution des classes. De même, quand ce déséquilibre est petit, la BNS est la seule métrique de sélection qui génère un meilleur résultat que celui avec l'utilisation de toutes les caractéristiques. Les SVM sont confirmés encore une fois comme le meilleur algorithme pour l'apprentissage.

Les performances du Bayésien Naïf et des SVM varient beaucoup en fonction des caractéristiques et des jeux de données utilisés. Wang et Manning [Wang and Manning, 2012] signalent que pour des tâches de classification de sentiments, un classifieur bayésien naïf dépasse les SVM sur de petits textes, et les SVM l'emportent à leur tour sur des documents plus longs.

Ces dernières années, l'outil Word2Vec inventé par Mikolov [Mikolov et al., 2013] a attiré l'attention de beaucoup de chercheurs dans la communauté de la recherche en fouille de textes. Grâce à cet outil, l'approche de l'apprentissage non supervisé a été bien plus utilisée dans les travaux de classification. Word2Vec peut réaliser une représentation vectorielle des mots ou des phrases selon leur contexte dans un corpus de grande volumétrie non annoté. Lin *et al.* [Lin et al., 2015] réalise une catégorisation de sentiments sur les commentaires des hôtels chinois par les internautes. Les caractéristiques d'entrée sont les textes vectorisés puis transformés par Word2Vec, sans l'appui d'aucun lexique de sentiment, puis la classification par un SVM est celle qui obtient les meilleures performances. Les travaux de Lilleberg *et al.* [Lilleberg et al., 2015] montrent que la combinaison de Word2Vec et du TF-IDF dépasse le TF-IDF seul, car Word2Vec permet de rajouter des caractéristiques complémentaires. Le but de Eensoo *et al.* [Eensoo et al., 2015] est d'analyser la subjectivité

dans les tweets. Ils confrontent la méthode textométrique combinée avec l'apprentissage supervisé aux représentations vectorielles. Ces dernières montrent des avantages lorsque les catégories sont nombreuses et le corpus d'entraînement est de taille limitée. Selon les résultats obtenus, combiner les deux permet d'obtenir des résultats très compétitifs.

Notre présente étude se concentre sur l'approche de l'apprentissage supervisé, et les quatre méthodes de sélection ont été adaptées afin de sélectionner les caractéristiques par classe. Diverses expériences ont été menées pour comparer les performances des classifieurs.

CHAPITRE 2

MÉTHODES DE SÉLECTION DE CARACTÉRISTIQUES IMPLÉMENTÉES

Sommaire

2.1 Spécificité lexicale

Dans les travaux de [Eensoo et al., 2015], pour extraire les critères linguistiques robustes qui serviront de fonctions caractéristiques aux méthodes d'apprentissage supervisé, une analyse sémantique est effectuée grâce au calcul de la spécificité lexicale. C'est une méthode de textométrie dont le calcul est proposé initialement par Lafon [Lafon, 1980] en appliquant la loi hypergéométrique à la question de la distribution de la fréquence des formes dans un corpus divisé en plusieurs fragments.

En fonction d'un seuil de probabilité choisi par l'analyste (généralement 0,05 comme le paramétrage par défaut de Lexico 3 [Salem et al., 2003]), l'indicateur probabiliste calculé permet de montrer si la fréquence d'un terme observée dans telle ou telle partie peut être normale ou non, et ainsi délimiter deux sous-ensembles de formes : celui des formes spécifiques à la partie considérée (positivement ou négativement), et celui des formes non-spécifiques représentant un signal statistiquement attendu. Les formes spécifiques indiquent un sur-emploi ou un sous-emploi des termes dans la partie par rapport à l'ensemble du corpus [Labbe and Labbe, 1997]. Ainsi cette analyse différentielle du corpus permet d'identifier à partir du lexique, les différences entre les partitions comparées, ce qui en facilite la qualification et l'illustration via des mots sélectionnés.

Les résultats de calcul de spécificité dans Lexico 3 est un nombre qui rend compte du degré de significativité de l'écart entre la fréquence observée et la fréquence attendue d'une forme, par exemple un score de spécificité de +5 indique que la fréquence observée est plus grande que celle attendue, et la probabilité que cette distribution de fréquence soit attestée par hasard est de l'ordre de 10^{-5}, beaucoup plus petit que le seuil 0,05, donc il existe un sur-emploi de ce terme dans la partie courante. Le même calcul est réalisé pour les spécificités négatives, ou les sous-emplois de mots dans la partie du texte étudiée.

Notre but étant de ne conserver que les caractéristiques discriminantes qui représentent bien la thématique, la spécificité lexicale est intuitivement choisie comme une méthode de sélection. Son algorithme est implémenté dans le programme interne, chaque caractéristique recevra un score par rapport à chaque classe.

Les paramètres utilisés :

T : la longueur du corpus entier

t_i : la longueur des documents de la classe i

f : la fréquence absolue d'une forme dans le corpus entier

f_i : la fréquence absolue d'une forme dans les documents de la classe i

Avec ces paramètres, on calcule une probabilité pour qu'une forme de fréquence f apparaisse f_i fois dans la partie i selon la formule de la loi hypergéométrique :

$$P(X = f_i) = \frac{\binom{f}{f_i}\binom{T-f}{t_i-f_i}}{\binom{T}{t_i}}$$

Un seuil est choisi pour filtrer les caractéristiques : garder celles dont le score est inférieur au seuil, et quand les spécificités négatives sont aussi prises en compte, comparer leur valeur absolue avec le seuil.

2.2 TF-IDF

Jing *et al.* [Jing et al., 2002] utilisent le TF-IDF pour filtrer les caractéristiques, à part les pré-traitements traditionnels, ils remplacent aussi les noms d'emplacement et de personne par « *Word_Place_Name* » et « *Word_Person_Name* » respectivement, pour réduire certaines opérations répétitives non nécessaires. Sur un corpus qui contient 135 catégories, leur exactitude de classification s'élève à 76%, qui est meilleur que la méthode IDF dont le résultat est 61%.

Le TF-IDF est un modèle de pondération proposé par Salton et Buckley [Salton and Buckley, 1988], qui est beaucoup utilisé en recherche d'information et en fouille de textes, qui estime l'importance d'un terme dans un document relativement à un corpus : le poids augmente proportionnellement à la fréquence d'apparition du terme dans le document, mais en même temps il est ajusté par la fréquence du mot dans le corpus. L'idée est que si un terme est présent dans de nombreux documents, il est en fait peu discriminant. En conséquence le schéma propose d'augmenter la pertinence d'un terme en fonction de sa rareté au sein du corpus, voici la formule (notons $tf_{i,j}$ la fréquence d'apparition d'un terme i dans le document j, N le nombre total des documents, et df_i le nombre de documents où le terme i figure au moins une fois) :

$$tf_{i,j} \times \log(\frac{N}{df_i})$$

Dans notre cas d'étude, pour choisir les meilleures caractéristiques classées par le TF-IDF pour chaque thématique, on a calculé le score par rapport à chaque classe au lieu de par rapport à chaque document :

fréquence d'un terme dans une classe :

$$\frac{\text{nombre d'occurrences d'un terme dans une classe}}{\text{nombre d'occurrences de tous les termes dans cette classe}}$$

fréquence inverse d'apparition d'un terme dans les classes :

$$\log_{10}(\frac{\text{nombre de classes}}{\text{nombre de classes où ce terme figure}})$$

La valeur du TF-IDF est toujours supérieure ou égale à 0. Lorsqu'elle est nulle, soit la caractéristique n'est pas présente dans cette classe, soit on la retrouve dans toutes les classes. Un seuil est donné, qui permet de sélectionner les caractéristiques dont le score est supérieur au seuil et ensuite de tester la performance de la classification avec ce filtrage des caractéristiques.

2.3 Information mutuelle

Dans la théorie des probabilités et la théorie de l'information, l'information mutuelle (MI) de deux variables aléatoires mesure leur dépendance statistique. La valeur est nulle si et seulement si ces deux variables sont indépendantes, et croît lorsque la dépendance augmente. La formule est la suivante :

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log(\frac{p(x,y)}{p(x)\times p(y)})$$

Dans les études de corpus linguistique, c'est la mesure PMI (*Pointwise Mutual Information*) qui est beaucoup utilisée [Yang and Pedersen, 1997, Sebastiani, 2002]. La formule de PMI entre deux évènements singuliers x et y est définie comme :

$$PMI(x,y) = \log(\frac{p(x\wedge y)}{p(x)\times p(y)})$$

PMI compare la probabilité d'observer t (un terme) et c (une classe) ensemble (probabilité jointe) avec celle d'observer t et c indépendamment.

— s'il existe une vraie association entre t et c, $p(t\wedge c)$ sera beaucoup plus grande que $p(t)\times p(c)$, $PMI(t,c)\gg 0$;
— s'il n'existe pas une relation significative entre t et c, $p(t\wedge c)\approx p(t)\times p(c)$, $PMI(t,c)\approx 0$;
— si t et c ont une corrélation négative, $p(t\wedge c)$ sera beaucoup plus petite que $p(t)\times p(c)$, qui force $PMI(t,c)\ll 0$.

En conséquence, la valeur de PMI peut être négative, par contre la valeur de MI de n'importe quelles variables X et Y est toujours non négative selon la théorie de l'information.

Il existe pourtant une faiblesse de PMI, comme ce que montre cette formule équivalente :

$$PMI(t,c) = \log(\frac{p(t\wedge c)}{p(t)\times p(c)}) = \log p(t|c) - \log p(t)$$

Le score est fortement influencé par la probabilité marginale d'un terme : à probabilité conditionnelle $p(t|c)$ égale, les termes rares auront un score plus grand que les termes communs. Donc les scores de PMI ne sont pas comparables à travers les termes de fréquence différente. En plus, il est sensible aux erreurs d'estimation de probabilité quand $p(t)$ est proche de zéro [Yang and Pedersen, 1997].

Xu *et al.* [Xu et al., 2007] appliquent une mesure dérivée de la définition originale de MI, notons $\{c_i\}_{i=1}^m$ l'ensemble des catégories, $C = \cup_{i=1}^m c_i$, et $T = \{t, \bar{t}\}$ l'ensemble où le terme apparaît ou pas, donc la formule entre T et C est définie comme ceci :

$$I(T;C) = \sum_{i=1}^{m} p(t \wedge c_i) \log_2(\frac{p(t \wedge c_i)}{p(t) \times p(c_i)}) + \sum_{i=1}^{m} p(\bar{t} \wedge c_i) \log_2(\frac{p(\bar{t} \wedge c_i)}{p(\bar{t}) \times p(c_i)})$$

Ils combinent les scores spécifiques aux classes en valeur moyenne ou maximale pour filtrer les caractéristiques, avec les classifieurs k-NN et Bayésien Naïf, cette mesure obtient un meilleur résultat que PMI sur le corpus Reuters-21578 et OHSUMED.

Dans notre étude, on a choisi PMI pour filtrer les caractéristiques pour chaque thématique, étant donné une table de confusion d'un terme t et d'une classe c (voir tableau 2.1), on note :

	c	$\neg c$
t	A	B
$\neg t$	C	-

TABLE 2.1 – Table de contingence d'un mot et d'une classe

A : le nombre de documents de classe c où t figure ;
B : le nombre de documents qui ne sont pas de classe c où t figure ;
C : le nombre de documents de classe c où t ne figure pas ;
N : le nombre total des documents dans le corpus.
Donc la valeur de PMI est estimée comme ceci :

$$PMI(t,c) \approx \log_2(\frac{(A \times N)}{(A+C) \times (A+B)})$$

Un seuil est fourni pour filtrer et garder les caractéristiques avec les scores supérieurs au seuil.

2.4 Différence Proportionnelle Catégorique

CPD (*Categorical Proportional Difference*) est une méthode proposée par Simeon et Hilderman [Simeon and Hilderman, 2008], qui est une mesure à laquelle une caractéristique contribue à différencier une catégorie particulière des autres catégories. Ils l'évaluent avec un classifieur SVM et un classifieur bayésien naïf sur des données extraites des corpus OHSUMED, 20 Newsgroups (20 classes) [Newsgroups, 1999] et Reuters-21578. En comparaison avec six autres méthodes courantes, selon la F-mesure, CPD obtient la meilleure performance dans quatre sur six tâches de classification.

Avec les paramètres dans le tableau 2.1, la valeur CPD pour un terme t dans une classe c est définie comme suit :

$$CPD(t,c) = \frac{A-B}{A+B}$$

Les valeurs possibles de CPD sont restreintes dans l'intervalle [-1,1] :

— CPD proche de -1 indique que le mot ne figure presque pas dans les documents d'une certaine classe ;
— CPD = 1 signifie que le mot ne figure que dans les documents d'une certaine classe.

La valeur CPD d'un terme est le ratio associé avec une classe où la CPD obtient la valeur maximale, soit :

$$CPD(t) = max_i\{CPD(t, c_i)\}$$

Donc dans la comparaison des méthodes de [Simeon and Hilderman, 2008], les caractéristiques sont filtrées globalement en convertissant les scores spécifiques aux classes aux scores maximaux sur toutes les classes. Par contre dans notre cas, à l'instar des trois méthodes présentées ci-dessus, on utilise un seuil pour filtrer les caractéristiques par rapport à chaque catégorie. Après des expériences, on décide de garder les caractéristiques dont la valeur absolue du score CPD est plus grande ou égale au seuil.

CHAPITRE 3

ALGORITHMES D'APPRENTISSAGE AUTOMATIQUE APPLIQUÉS

Sommaire

3.1 Bayésien Naïf Multinomial

L'algorithme principal de classification que nous avons appliqué est un classifieur bayésien naïf multinomial, implémenté dans Weka [Hall et al., 2009]. Le mot « naïf » dans son nom signifie qu'il impose une hypothèse simplificatrice dans les interactions entre les caractéristiques. En tant qu'un modèle probabiliste, pour un document donné (noté d), parmi toutes les classes $c \in C$, le classifieur va retourner la classe (noté $\widehat{c}$) qui détient la plus grande probabilité de prédiction :

$$\widehat{c} = argmax_{c \in C} P(c|d)$$

Le travail de Bayes [Bayes, 1763] est à l'origine de l'inférence bayésienne, et Mosteller et Wallace [Mosteller and Lee Wallace, 1964] l'ont appliqué pour la première fois à la tâche de classification.

Selon le théorème de Bayes, voici la relation entre la probabilité jointe et la probabilité conditionnelle de deux variables A et B :

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Ainsi notre formule de prédiction est égale à :

$$\widehat{c} = argmax_{c \in C} P(c|d) = argmax_{c \in C} \frac{P(d|c)P(c)}{P(d)}$$

L'objectif est de comparer pour un document donné cette probabilité pour chaque classe possible, donc on peut ignorer le $P(d)$ vu que c'est toujours la même.

$$\widehat{c} = argmax_{c \in C} P(c|d) = argmax_{c \in C} P(d|c)P(c)$$

Dans cette formule simplifiée, $P(c)$ s'appelle la probabilité a priori de la classe c, $P(d|c)$ s'appelle la vraisemblance du document d, et $P(c|d)$ est la probabilité a posteriori que l'on cherche à calculer.

Dans notre cas, un document peut être représenté par un ensemble de caractéristiques :

$$\widehat{c} = argmax_{c \in C} P(t_1, t_2, \ldots, t_n|c)P(c)$$

Or il est toujours très difficile de calculer la probabilité de toutes les possibilités de combinaison de toutes ces caractéristiques car cela nécessiterait énormément de données et de calculs. Par conséquent, le modèle Bayésien Naïf impose deux suppositions :

— premièrement c'est l'hypothèse du « sac de mots » : chaque caractéristique possède le même effet de classification peu importe sa position dans le document, seule sa fréquence est prise en compte ;
— la deuxième est généralement appelée « hypothèse bayésienne naïve » qui suppose la forte indépendance entre les caractéristiques, donc les probabilités conditionnelles $P(t_i|c)$ peuvent être naïvement multipliées :

$$P(t_1, t_2, \ldots, t_n|c) = P(t_1|c) \times P(t_2|c) \times \cdots \times P(t_n|c)$$

Ceci fait que la formule finale de la classe estimée par le classifieur est (notons T pour l'ensemble de toutes les caractéristiques différentes) :

$$\widehat{c} = argmax_{c \in C} P(c) \prod_{t_i \in T} P(t_i|c)$$

Cependant, le produit de nombreuses probabilités conditionnelles inférieures à 1 peut rapidement provoquer des débordements de capacités de représentation de nombres flottants. Pour contourner ce problème et augmenter la vitesse de calcul, nous passons traditionnellement par les logarithmes lors de la mise en œuvre de la méthode :

$$\widehat{c} = argmax_{c \in C} \log P(c) + \sum_{t_i \in T} \log P(t_i|c)$$

Le classifieur bayésien naïf est un classifieur linéaire tel que le montre la formule, la classe prévue est calculée comme une fonction linéaire sur des caractéristiques données.

En utilisant le maximum de vraisemblance pour l'estimation, la probabilité a priori $P(c)$ est le pourcentage des documents de la classe c dans le corpus. Et la vraisemblance $P(t_i|c)$ est la probabilité qu'un document de classe c contienne le mot t_i.

En revanche, il existe un problème pour la vraisemblance. Il arrive que l'on la calcule pour une caractéristique qui n'est jamais présente dans des documents d'une certaine classe c, dans ce cas-là $P(t_i|c)$ sera 0. Mais comme la formule multiplie naïvement toutes les vraisemblances ensemble, cela conduira à une valeur 0 pour toute la classe quelle que soit la probabilité a priori.

Pour pallier à ce problème, on a choisi la méthode de lissage Laplace qui transforme le calcul en :

$$P(t_i|c) = \frac{nbOccurrence(t_i, c) + 1}{\sum_{t \in T}(nbOccurrence(t, c) + 1)} = \frac{nbOccurrence(t_i, c) + 1}{(\sum_{t \in T} nbOccurrence(t, c)) + |T|}$$

Dans notre cas de classification multi-classe où toutes les classes s'excluent mutuellement, un classifieur binaire est construit pour chaque classe sur les exemples

de cette classe (positifs) et tous les autres exemples (négatifs). Pour un document de test, chaque classifieur est appliqué dessus et la classe avec la plus grande probabilité a posteriori est choisie comme résultat de prédiction.

La représentation du document est une étape importante dans la classification pour faciliter le traitement des classifieurs, ici c'est le vecteur de caractéristiques qui est utilisé, et le modèle multinomial peut gérer le cas où les caractéristiques ont pour poids le nombre d'occurrences dans le document courant.

3.2 Machines à Vecteurs de Support (SVM)

Les SVM ont été développés dans les années 1990, basé sur le principe de la « minimisation structurelle du risque » [Vapnik, 1995], et ont été introduits dans la classification des textes par Joachims [Joachims, 1998].

A partir d'un ensemble de données d'apprentissage annotées, l'algorithme des SVM cherche à trouver un hyperplan optimal pour catégoriser ces données connues et d'éventuelles nouvelles données. Si on simplifie le problème en une classification de points linéairement séparables dans une surface donnée (voir figure 3.1), il existe plusieurs lignes droites qui peuvent séparer les données en deux classes. Mais intuitivement on sait qu'une ligne très proche des données est sensible aux bruits et sera mal généralisée. Entre l'hyperplan et les données de classes différentes les plus proches de lui, ici représentées par le carré rouge et le rond bleu sur les pointillés, il existe une distance minimale de séparation. L'algorithme des SVM vise à trouver l'hyperplan qui rend cette distance la plus grande.

La « marge maximale » est deux fois cette distance maximale, et les données sur les pointillés s'appellent les « vecteurs de support », qui sont composés seulement d'une petite partie des exemples d'apprentissage. Cette notion simplifiée s'applique tout à fait aux cas des grandes dimensions, où la ligne droite simplifiée deviendra un vrai hyperplan c'est-à-dire une surface de décision.

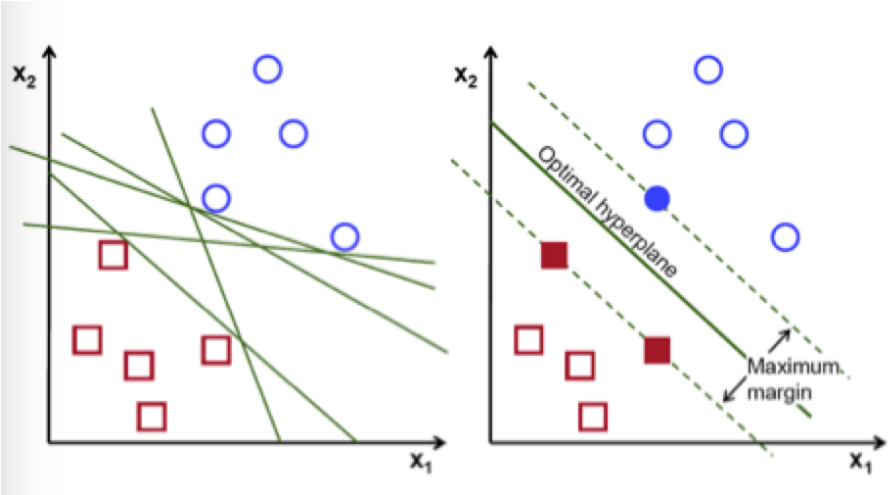

FIGURE 3.1 – SVM cherchent à trouver l'hyperplan optimal

Cet algorithme est aussi applicable aux données non linéairement séparables, selon les expériences de Yang et Liu [Yang and Liu, 1999], les résultats sont légèrement meilleurs pour les données linéairement séparables. Selon les arguments dans

[Joachims, 1998], la sélection des caractéristiques n'est généralement pas nécessaire pour les SVM vu qu'ils sont assez robustes pour contrôler le sur-apprentissage grâce à leurs paramètres de régularisation. En plus, d'après Joachims, les SVM possèdent une bonne adaptabilité quand on passe à une grande échelle de dimensionnalité, parce que la mesure de la complexité des hypothèses est basée sur la marge maximale qui sépare des données, et non sur le nombre de caractéristiques.

En utilisant les SVM, on cherche en fait un compromis entre un hyperplan avec la marge maximale et un hyperplan qui peut classifier le plus que possible des données d'apprentissage. Pour cela, le paramètre C permet d'ajuster une tolérance de classifications erronées d'exemples d'apprentissage. Avec une large valeur de C, l'optimisation va choisir un hyperplan de plus petite marge, si cet hyperplan peut classifier correctement tous les exemples d'apprentissage au profit de la performance de généralisation. Et inversement, une petite valeur de C va conduire à un hyperplan avec une plus grande marge. Même si cette dernière provoque des données d'apprentissage faussement classées, on aura une meilleure performance de généralisation.

Étant un classifieur binaire, les SVM s'appliquent aux tâches de classification multi-classe par deux approches : 1) *1-against-the rest*, pour avoir m-classe classifieurs, construire m classifieurs binaires qui séparent chaque classe de toutes les autres classes 2) *1-against-1*, construire un classifieur binaire pour chaque paire possible de classes, soit $\binom{m}{2} = \frac{(m-1)\times m}{2}$ classifieurs binaires. Dans notre travail, on a utilisé la librairie LIBSVM [Chang and Lin, 2011] qui implémente la deuxième approche.

LIBSVM demande un format de corpus d'entrée différent de celui de ARFF dans Weka (voir section 4.3), qui représente chaque document comme suit :

$$< catégorie > < index1 >:< valeur1 > < index2 >:< valeur2 > \ \ldots$$

La catégorie est celle annotée préalablement, le couple index et valeur signifie l'index de la caractéristique dans la liste des attributs et sa valeur correspondante.

On a réalisé une sélection de caractéristiques simple avec les configurations dans le tableau 5.2, soit sans aucune méthode de sélection prédéfinie, puis transformé les fichiers ARFF en format qui convient au LIBSVM. Après des expériences préliminaires, on a trouvé que la fonction de noyau linéaire convient bien à nos données. Les résultats de la validation croisée sont présentés dans la section 5.8.

CHAPITRE 4

CORPUS, PARAMÈTRES ET ALGORITHME

Sommaire

4.1 Caractéristiques et statistiques du corpus

A la différence des corpus de référence utilisés dans des travaux précédents, notre corpus français est constitué d'articles de blogs et d'éditoriaux issus des différents types de sites. L'annotation des classes n'est pas manuelle car ceci aurait demandé un coût humain et temporel trop élevé pour annoter et régler les conflits entre annotateurs. Il existe aussi des bouts de codes de JavaScript ou HTML comme bruit.

Le crawler interne de XiKO récupère les données textuelles du web grâce à des fichiers de configuration pour chaque nom de domaine. Une information importante est le fil d'ariane, qui correspond à la catégorie de la page courante choisie par les éditeurs des sites. Par exemple quand on rencontre « Annonces automobiles > Citroen > C4 > THP 155 EXCLUSIVE BMP6 », la page va être annotée avec la catégorie interne « *xiko:topic.automotive* ». Un fichier gère ces associations par les moyens d'expressions régulières, regroupées selon les noms de domaine, dont les entrées sont sous la forme comme suit :

*<map territory="maman.*conception" topic="xiko:topic.health_well_being-pregnancy_birth"/>*

ou bien quand des URLs sont déjà significatifs, l'annotation peut aussi s'appuyer dessus :

<map url="./agriculture-alimentation/Le-Vin.*" topic="xiko:topic.food_drink-alcohol-wine"/>*

Pour des thématiques fines assez similaires, un fil d'ariane ou URL peut recevoir plusieurs annotations, par exemple :

<map territory="grossesse-bébé" topic="xiko:topic.family_parenting-babies_toddlers"/>

<map territory="grossesse-bébé" topic="xiko:topic.health_well_being-pregnancy_birth"/>

Notre système d'annotation exploite le travail humain des éditeurs des sites en vue de générer les données de référence pour la classification ultérieure. Donc ce fichier de paramètre est mis à jour soigneusement pour contrôler au maximum les ambiguïtés. On est aussi conscient des différences entre les rubriques choisies et les réalités du terrain, par exemple la rubrique « société » est souvent un amalgame des articles sur des différents sujets plus fins, qui peut présenter des difficultés pour la classification.

Les documents crawlés disposant de ces renseignements vont être ainsi annotés automatiquement. Après ils seront stockés dans une base de données et ils peuvent être exploités à travers les requêtes de *BigQuery* [1] de Google. Après un échantillonnage équilibré et un dédoublonnage, le corpus utilisé dans ce travail est composé de 159 658 documents.

Les 27 classes de référence se trouvent au premier niveau dans la taxonomie des segments chez XiKO. Une thématique de deuxième niveau dans la hiérarchie est sous la forme « *xiko:topic.health_well_being-psychology* » où « *health_well_being* » est la catégorie première et « *psychology* » lui est lié avec un tiret qui signifie que c'est une sous-thématique plus fine. Or, pour nos expériences, nous ne chercherons qu'à détecter les thématiques principales. Pendant le processus, les annotations ont été nettoyées selon un fichier de paramètre, en vue de :

— ne garder que les thématiques de premier niveau, par exemple « *xiko:topic.health_well_being-psychology* » a été remplacé par « *xiko:topic.health_well_being* » ;
— ignorer les classes où les documents sont trop peu présents, par exemple « *xiko:topic.video_computer_games* » ou « *xiko:topic.food_drink* » qui possède seulement une trentaine de documents ;
— équilibrer la distribution des classes dans le corpus, par exemple ne pas prendre en compte les documents des classes « *xiko:topic.news* » et « *xiko:topic.news-international* » qui détiennent 22 701 documents en total.

Après ce nettoyage de corpus, il reste 136 821 documents dans le corpus. Les textes ont une longueur moyenne de 370 tokens, et la distribution des classes est montrée dans le tableau 4.1.

4.2 Paramètres du classifieur

Tout le processus est réalisé en Java. Dans la ligne de commande, on donne l'option si on veut valider la classification à la façon croisée en dix plis, avec la sélection de caractéristiques effectuée sur chaque sous-échantillon d'apprentissage avant de construire le modèle. Tous les autres paramètres sont gérés dans un fichier de propriété, qui sont explicités dans le tableau 4.2.

1. https://cloud.google.com/bigquery/

Nom de la catégorie	Nombre de documents	Pourcentage de la catégorie
sports	32 428	23,70%
culture & entertainment	20 513	14,99%
health & well-being	9 931	7,26%
business	7 814	5,71%
politics	7 666	5,60%
technology	6 747	4,93%
society	6 716	4,91%
news	6 456	4,72%
science	5 158	3,77%
environment	4 049	2,96%
travel	3 332	2,44%
personal finance	3 096	2,26%
cooking	2 876	2,10%
style & fashion	2 833	2,07%
home & garden	2 726	1,99%
family & parenting	2 233	1,63%
pets	2 020	1,48%
beauty	1 787	1,31%
legal issue	1 785	1,30%
real estate	1 698	1,24%
education	1 260	0,92%
weddings	880	0,64%
outing	691	0,51%
automotive	682	0,50%
spirituality	574	0,42%
dating & love & couple	474	0,35%
careers	396	0,29%

TABLE 4.1 – Distribution des classes dans le corpus

4.3 Algorithme

La validation croisée est une méthode qui évalue la capacité d'une méthode d'apprentissage à produire des modèles généralisés, qui consiste à diviser les données en k parties (ici 10), chaque fois une partie est sélectionnée comme échantillon de validation, et les autres $(k-1)$ parties sont utilisées comme données d'apprentissage. On répète cette opération k fois en choisissant chaque partie juste une seule fois pour évaluer (voir figure 4.1).

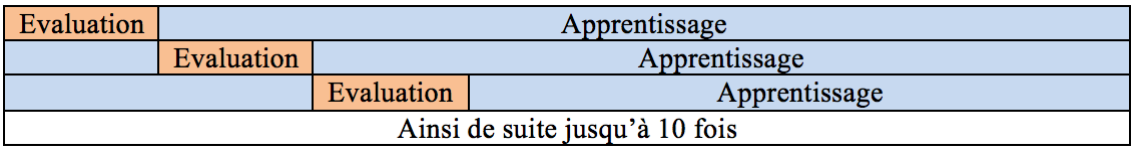

FIGURE 4.1 – Schéma de la validation croisée

Le but de la validation croisée est de valider la robustesse du processus d'appren-

tissage, pour éviter le « coup de chance » (ou de malchance) qui pourrait faire que les résultats soient localement bons (ou mauvais). Cependant, si on effectue la sélection de caractéristiques une seule fois sur toutes les données puis on fait la validation croisée, pour chaque sous-échantillon testé, le fait que l'on a sélectionné les caractéristiques sur l'ensemble d'apprentissage et aussi celui de test fausse l'évaluation. C'est pour pallier ce problème que l'on a implémenté un algorithme où on intègre la sélection de caractéristiques dans la validation croisée, c'est-à-dire qu'on répète 10 fois la sélection de caractéristiques sur chaque sous-échantillon d'apprentissage avant de construire le modèle.

Prétraitement du corpus : nous avons minusculisé le corpus et n'avons conservé que les caractères alphanumériques. La racinisation est optionnelle et est effectuée par SnowBall [Porter, 2001]. Un corpus lemmatisé a été généré à l'aide de l'outil TreeTagger [2] [Schmid, 1994].

Notre algorithme est décrit dans la figure 4.2.

Pour chaque document dans le corpus :

Remplir le jeu de données du corpus entier (*trainSet*)

Réaliser une répartition aléatoire des documents dans 10 parties

Parcourir une deuxième fois le corpus, et pour chaque document :

Selon l'association produite (index du document → numéro de partie), construire 10 échantillons d'évaluation

Pour chacun des 10 échantillons d'évaluation :

Construire les échantillons d'apprentissage correspondant (à l'aide de *trainSet*)

Filtrer les caractéristiques par les sélecteurs au niveau global => *setA*

Sur la base des caractéristiques dans *setA*, pour chaque classe, sélectionner les caractéristiques par les sélecteurs locaux

Regrouper ces caractéristiques locales, construire un modèle Bayésien Naïve Multinomial pour chaque partie d'apprentissage, puis l'évaluer sur la partie d'évaluation correspondante (étape à répéter quand on affine le modèle)

Calculer les mesures d'évaluation, la matrice de confusion ainsi que le nombre de caractéristiques utilisées en moyenne

FIGURE 4.2 – Algorithme de la validation croisée avec la sélection de caractéristiques intégrée

Les caractéristiques sélectionnées deviennent des attributs dans les fichiers en format ARFF de Weka [Hall et al., 2009], où chaque document est représenté sous le format « sparse », par exemple :

2. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

{2073 1,3348 1,3669 1,5144 1,9287 1,13449 1,49260 "$xiko$:topic.sports"}

Pour chaque couple de données, le premier nombre est l'index d'une caractéristique dans la liste des attributs, et le deuxième est le nombre d'occurrences de cette caractéristique dans le document courant. Les caractéristiques absentes ne sont pas prises en compte dans le format « sparse ». La catégorie de référence est mise à la fin avec son index dans la liste des attributs.

Pour chaque catégorie, une méthode de sélection prédéfinie attribue à chaque caractéristique un score par rapport à cette catégorie. Selon un seuil donné, seules les caractéristiques sélectionnées sont considérées spécifiques à cette classe. Elles sont ordonnées et écrites dans des fichiers séparés pour faciliter l'analyse ultérieure.

Pour chaque sous-échantillon d'évaluation, un fichier est généré où les informations suivantes sont renseignées pour chaque document :

Résultat de classification (succès ou échec) ; catégorie de référence ; catégorie de prédiction ; liste des caractéristiques choisies qui sont apparues dans ce document

On exploite ces résultats de l'évaluation pour affiner le modèle. Le but est d'exclure les caractéristiques qui figurent dans le plus de cas d'échec, donc on calcule le ratio de génération d'erreur pour chaque caractéristique présente dans les documents de test :

$$\frac{\text{nombre de documents mal classés}}{(\text{nombre de documents mal classés} + \text{nombre de documents bien classés})}$$

Le seuil de ratio étant fixé à 90%, en supprimant les caractéristiques dont le ratio d'erreur dépasse ce seuil, on reconstruit le modèle et réévalue.

Au niveau de la segmentation pour construire des jeux de données, pour les caractéristiques en unigrammes, au vu de l'hypothèse d'indépendance forte de Bayésien Naïf, on perd de l'information apportée par les collocations par rapport aux mots simples, qui pourrait aider à améliorer la classification, par exemple « produit cosmétique », « syndrome respiratoire aigu » ou bien « collection haute couture ».

Dans la recherche de [Aït-Hamlat, 2016], un modèle bayésien multinomial est construit de façon incrémentale et itérative sur une vaste base de données évolutive pour prédire deux indicateurs socio-démographiques « âge » et « genre », où la segmentation en n-grammes conduit à une bonne performance de classification. Fürnkranz [Fürnkranz, 1998] utilise l'algorithme d'apprentissage de règles RIPPER pour tester la force de n-grammes sur le corpus 20 NewsGroups et Reuters-21578. Afin d'éviter la croissance linéaire du nombre de n-grammes générés, il implémente un algorithme de génération de caractéristiques basé sur l'algorithme APRIORI [AGRAWAL et al., 1995]. Des bigrammes ou trigrammes s'avèrent les plus performants après un filtrage par un seuil de nombre d'occurrences minimum modéré, et l'utilisation de séquences plus longues dégrade la performance.

De notre côté, on a segmenté le corpus par le $NGramTokenizer$ de Weka pour tester la performance des caractéristiques en bigramme et trigramme. Les résultats sont présentés dans la section 5.5.

Nom de paramètre	Type de valeur	Rôle du paramètre
Nettoyage des annotations		
topicMap	path	chemin vers le fichier de paramètre qui sert à nettoyer les annotations
Normalisation du corpus		
lowercase	booléen	minusculiser les mots
stemmatize	booléen	raciniser les mots
removeAccent	booléen	désaccentuer les mots
Sélecteurs au niveau global (appliqués dans cet ordre)		
formMinOccurrence	entier	nombre d'occurrences minimum d'un terme dans le corpus d'apprentissage pour être considéré comme un candidat
tokenMinLength	entier	longueur minimum d'un terme pour être considéré comme un candidat
removeDigit	booléen	ignorer les chiffres et nombres
removeStopWords	booléen	ignorer les mots-outils (selon une liste de 215 mots-outils français)
Sélecteurs par rapport à chaque classe (les 4 premiers appliqués dans cet ordre)		
ratioInPart	double	ratio d'occurrences minimum d'un terme dans une classe pour être considéré comme un candidat
formMinOccurrenceInPart	entier	nombre d'occurrences minimum d'un terme dans une classe pour être considéré comme un candidat
methodName_threshold	double	seuil aux méthodes de sélection qui filtre les caractéristiques pour chaque classe
nbFeatureByTopic	entier	nombre des caractéristiques les mieux classés à retenir pour chaque classe
sortByComputation	chaîne de caractères	selon la méthode appliquée, ordonner les caractéristiques différemment
negativeSpec	booléen	prendre en compte les mots avec une spécificité négative
Affinage du modèle		
excludedTopErrorFeatures	booléen	une fois le modèle évalué, exclure les caractéristiques générant le plus d'erreurs puis reconstruire le modèle et réévaluer
ratioFeatureExcluded	double	si une caractéristique apparaît dans ce ratio de cas d'échec de classification, exclure-le puis reconstruire le modèle

TABLE 4.2 – Paramètres qui composent la configuration

CHAPITRE 5

EXPÉRIENCES ET RÉSULTATS

Sommaire

Des mesures d'évaluation classiques sont calculées pour chaque classe :

$$\text{Précision} = \frac{\text{nombre de documents correctement classifiés d'une classe c}}{\text{nombre de documents qui ont été classifiés dans cette classe c}}$$

$$\text{Rappel} = \frac{\text{nombre de documents correctement classifiés d'une classe c}}{\text{nombre de documents qui ont été annotés dans cette classe c}}$$

$$\text{F-mesure (moyenne harmonique de précision et rappel)} = \frac{2 \times \text{précision} \times \text{rappel}}{\text{précision} + \text{rappel}}$$

Ainsi que des mesures qui évaluent la performance sur l'ensemble des classes :

Micro-moyenne F-mesure : faire la somme des valeurs les plus fines, puis calculer la mesure, ce qui est pondéré par la distribution des classes.

Macro-moyenne F-mesure : moyenne arithmétique de toutes les F-mesure, qui donne un poids égal à toutes les classes, donc elle est plus pertinente quand les performances sur chaque classe sont équitablement importantes [Jurafsky and Martin, 2015, chapitre 7].

Avant toutes les comparaisons, l'exactitude de la classification sans aucune règle, soit quand le système met tous les documents dans la classe détenant le plus de documents (ici *Sports*), est de 23,70%.

5.1 Sans application de méthode de sélection

Pour savoir quelles sont les performances obtenues sans aucune méthode de sélection (après filtrage de nombres et de mots-outils, ainsi que des mots avec une fréquence basse), le modèle est construit sur le corpus minusculisé et non racinisé. Les résultats sont présentés dans le tableau 5.1.

	MinOccurrence = 10	MinOccurrence = 3	MinOccurrence = 0
Nombre de caractéristiques en moyenne	81 710	158 216	320 787
Micro-moyenne F-mesure	72,15%	72,26%	-
Macro-moyenne F-mesure	62,30%	59,32%	-

TABLE 5.1 – Résultats de classification sans méthode de sélection

On voit qu'en supprimant plus de mots de fréquence basse, qui sont considérés peu utiles pour la classification à cause de leur rareté [Forman, 2003], on obtient un gain en macro-moyenne F-mesure. Ce qui signifie qu'il y a plus de documents correctement classifiés dans les petites classes. C'est le phénomène inverse pour la micro-moyenne : filtrer plus de caractéristiques semble pénaliser la classification pour les grandes classes. Si on n'effectue aucune sélection, le nombre de caractéristiques en moyenne s'élève à 320 787, à cause de cette grande dimension, après avoir construit le modèle sur la septième partie d'apprentissage, la machine virtuelle de Java n'a plus de mémoire donc on n'a pas eu de résultat d'évaluation. En plus, le temps utilisé est beaucoup plus long que celui avec l'utilisation des méthodes de sélection, ce qui n'est pas du tout efficace pour l'entreprise.

5.2 Comparaison entre les méthodes de sélection

Les expériences sont menées avec ces configurations basiques (voir tableau 5.2), les résultats avec la racinisation vont être présentés dans la section 5.4.1.

minimum nombre d'occurrence	minusculiser	ignorer nombres	ignorer mots-outils	raciniser	désaccentuer
3	oui	oui	oui	non	non

TABLE 5.2 – Configurations basiques générales

Pour trouver le meilleur seuil de score de chaque méthode, on a retenu pour chaque classe les 1000 premières caractéristiques les mieux classées (i.e. le paramètre *nbFeatureByTopic*). Après des expériences, les meilleurs résultats sont obtenus avec les seuils suivants : spécificité lexicale (0.00001), CPD (0.99), TF-IDF (0.00), et information mutuelle (1.00). Toutes les expériences suivantes ont utilisé ces seuils.

Le meilleur seuil de score étant fixé, on a varié le paramètre *nbFeatureByTopic* pour voir leur performance quand le nombre de caractéristiques utilisées augmente

(voir figure 5.1). A travers ces quatre graphes, on voit que le CPD arrive à filtrer le plus de caractéristiques quand il n'y a plus de seuil de *nbFeatureByTopic*, et ses performances varient beaucoup selon le nombre de caractéristiques utilisées. Tandis que la spécificité lexicale est plus stable que le CPD, avec un bien meilleur résultat au début et moins de caractéristiques supprimées à la fin. L'information mutuelle est très sensible au nombre de caractéristiques utilisées, et à la fin le TF-IDF obtient les meilleures performances avec le moins de caractéristiques au début, et ensuite reste stable. Sans le seuil *nbFeatureByTopic*, le TF-IDF a gardé beaucoup plus de caractéristiques comme l'information mutuelle, et la macro-moyenne F-mesure a baissé, ce qui signifie qu'il y a eu du bruit ajouté que l'algorithme ne parvient pas à détecter.

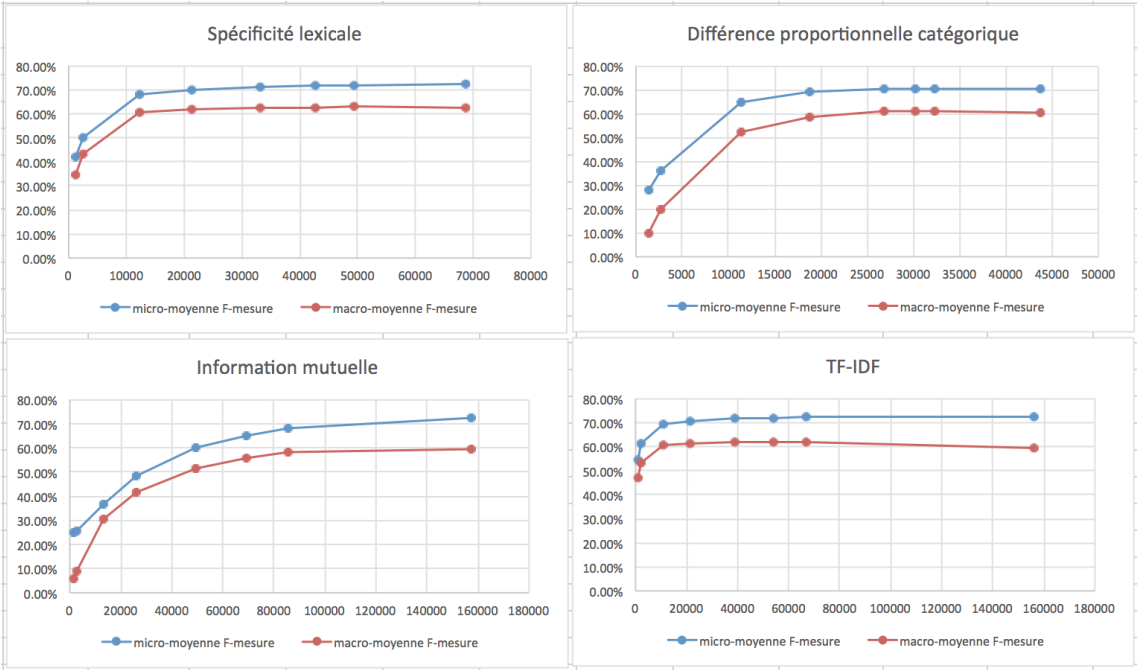

FIGURE 5.1 – Performance des quatre méthodes avec les meilleurs seuils, le paramètre *nbFeatureByTopic* contient la valeur de 50, 100, 500, 1000, 2000, 3000, 4000 et jusqu'à sans limite. L'axe horizontal représente le nombre de caractéristiques utilisées en total.

Quand on supprime le seuil *nbFeatureByTopic*, le nombre de caractéristiques choisies par classe dépend seulement de leur seuil de score, les résultats sont affichés dans le tableau 5.3.

On voit que l'information mutuelle obtient la meilleure micro-moyenne F-mesure mais ses caractéristiques utilisées sont les plus nombreuses. Sa macro-moyenne F-mesure est basse et elle donne de mauvais résultats sur des classes difficiles à classifier (*careers, dating_love_couple, outing*), où on a seulement des centaines de documents d'apprentissage. La meilleure macro-moyenne F-mesure avec des bons résultats sur les classes délicates sont obtenus par la spécificité lexicale, dont la micro-moyenne F-mesure est aussi satisfaisante. Nous remarquons surtout qu'elle réussit à faire beaucoup baisser le nombre de caractéristiques. Le CPD a utilisé encore moins de caractéristiques, avec des résultats un peu moins bons que ceux de la spécificité lexicale. Enfin le TF-IDF n'a pas filtré énormément de caractéristiques, ce qui apporte une très bonne micro-moyenne F-mesure mais la moins bonne macro-moyenne

Méthode	Spécificité lexicale, negativeSpec=false	CPD	TF-IDF	Information mutuelle
Nombre de caractéristiques en moyenne	68 618	43 762	156 404	157 267
Micro-moyenne F-mesure	72,14%	70,64%	72,18%	72,43%
Macro-moyenne F-mesure	62,46%	60,76%	59,17%	59,72%
F-mesure sur des classes difficiles (*careers, datingLoveCouple, outing*)	19,43%, 44,85%, 33,40%	17,16%, 45,18%, 32,67%	7,22%, 25,08%, 22,50%	1,94%, 20,31%, 22,70%

TABLE 5.3 – Comparaison des quatre méthodes avec leur meilleur seuil, sans la limite de *nbFeatureByTopic*

F-mesure.

5.3 Comparaison entre sans et avec méthode de sélection

Avec la configuration dans le tableau 5.2 et sans le seuil *nbFeatureByTopic* (un filtrage basique des caractéristiques dont le nombre d'occurrences est inférieur à 3 est toujours appliqué), les résultats dans le tableau 5.4 montre l'efficacité et l'utilité de notre travail de sélection de caractéristiques.

La spécificité lexicale a réussi de réduire 57% de caractéristiques, en gardant un résultat très satisfaisant de classification, et surtout en affichant les meilleures performances sur des classes difficiles. Le CPD a filtré encore plus de caractéristiques (réduction de 72%), mais ses résultats sont nettement inférieurs à ceux de la spécificité lexicale. Une réduction de caractéristiques selon une méthode adéquate renforce l'efficacité du processus et ne dégrade pas voire améliore les résultats du point de vue global, ceci confirme notre hypothèse de départ.

5.4 Effet de racinisation et lemmatisation

La racinisation et la lemmatisation sont considérées comme une forme de transformation de caractéristiques. La racinisation consiste à transformer des mots avec flexions en racine, en supprimant les préfixes et suffixes. L'algorithme du racinisateur SnowBall [Porter, 2001] que l'on utilise ici se compose d'une cinquantaine de règles

	Sans méthode de sélection prédéfinie	Spécificité lexicale, negativeSpec=false	CPD
Nombre de caractéristiques en moyenne	158 216	68 618	43 762
Micro-moyenne F-mesure	72,26%	72,14%	70,64%
Macro-moyenne F-mesure	59,32%	62,46%	60,76%
F-mesure sur des classes difficiles (*careers, datingLoveCouple, outing*)	1,47%, 17,02%, 23,69%	19,43%, 44,85%, 33,40%	17,16%, 45,18%, 32,67%

TABLE 5.4 – Comparaison de résultats entre sans et avec méthode de sélection

de racinisation classées en sept phases successives. Les mots à analyser passent par tous les stades et, dans le cas où plusieurs règles pourraient leur être appliquées, c'est toujours celle comprenant le suffixe le plus long qui est choisie. La racine peut correspondre à un mot réel de la langue ou non.

La lemmatisation d'une forme d'un mot consiste à en prendre sa forme canonique, qui désigne la forme à l'infinitif pour un verbe, et le mot au masculin singulier pour les autres mots. Un lemme correspond à un mot réel de la langue et constitue une entrée de dictionnaire. Ainsi après le prétraitement de la racinisation ou lemmatisation, le nombre des mots distincts diminue, qui permet de s'éloigner de la forme de surface des mots.

5.4.1 Racinisation

Avec les mêmes configurations que celles de la section précédente (sans le seuil *nbFeatureByTopic*, et voir tableau 5.2), sauf que les mots sont racinisés, les résultats sont présentés dans le tableau 5.5.

Les résultats sur le corpus racinisé sont similaires à ceux sur le corpus brut, mais le nombre de caractéristiques a été beaucoup réduit (19% $\sim$ 31%), d'où résulte moins de temps pour la construction de modèle.

5.4.2 Lemmatisation

Le corpus lemmatisé est construit à l'aide de TreeTagger, et il existe trois cas où les lemmes ne peuvent pas être récupérés directement dans la sortie de l'outil :

— si l'outil attribue l'étiquette <unknown> à un token, on reprend la forme initiale comme son lemme
— si le lemme est sous forme « payer | payer », on garde une fois le lemme
— si l'outil donne deux lemmes différents, par exemple « frai | **frais** » ou « **convenir** | convier » où seulement la forme en gras est le lemme correct en contexte, mais sa position est non prévisible, on garde les deux lemmes

	Sans méthode de sélection prédéfinie	Spécificité lexicale, negativeSpec=false	CPD	TF-IDF	Information mutuelle
Nombre de caractéristiques en moyenne	107 566	50 588	35 163	105 777	107 097
Micro-moyenne F-mesure	71,80%	71,60%	70,35%	71,35%	71,91%
Macro-moyenne F-mesure	59,54%	62,32%	60,79%	57,95%	59,77%

TABLE 5.5 – Résultats après le prétraitement de racinisation

	Sans méthode de sélection prédéfinie	Spécificité lexicale, negativeSpec=false	CPD	TF-IDF	Information mutuelle
Nombre de caractéristiques en moyenne	26 493	8 733	5 112	26 376	26 422
Micro-moyenne F-mesure	57,87%	57,03%	47,11%	57,97%	58,07%
Macro-moyenne F-mesure	43,16%	45,69%	32,75%	43,75%	43,85%

TABLE 5.6 – Résultats après le prétraitement de lemmatisation

Sur le corpus lemmatisé, il y a eu environ une baisse d'un ordre de grandeur de nombre de caractéristiques, et les résultats ont beaucoup dégradé par rapport au corpus brut, ce qui signifie une grande perte d'information (voir tableau 5.6). Comparons les 20 premières caractéristiques choisies pour la classe *cooking* par la spécificité lexicale (voir figure 5.2), la dernière colonne montre qu'après la lemmatisation, les meilleures caractéristiques sélectionnées sur le corpus brut sont classées vers le bas, ou sont disparues dans toute la liste.

5.5 Caractéristiques en n-grammes

On a filtré les n-grammes qui contiennent des mots-outils, des nombres ou des symboles. Sous les mêmes configurations (voir tableau 5.2), les caractéristiques ont été sélectionnées par la spécificité lexicale, sans le seuil *nbFeatureByTopic* ni la prise

1	Corpus lemmatisé	Corpus brut	
2	taillez	pâté	303
3	lavez	beurrés	91
4	sauce	cornets	disparue
5	saumon	mouillettes	disparue
6	tiédir	gla	disparue
7	réservez	whiskies	disparue
8	asperge	enlevez	disparue
9	tailler	grappes	disparue
10	cube	épongez	disparue
11	rôtir	beurrée	disparue
12	marinade	genviron	disparue
13	soigneusement	wine	disparue
14	amande	spéculoos	20
15	passez	perbellini	disparue
16	levure	gms	disparue
17	crabe	jayer	disparue
18	marron	xérès	disparue
19	tomate	cuisson	25
20	spéculoos	taboulé	479
21	gousse	passard	disparue

FIGURE 5.2 – Les 20 premières caractéristiques choisies pour la classe *cooking*

en compte des caractéristiques avec des spécificités négatives. Les n-grammes dont le nombre d'occurrences est inférieur à 3 sont filtrés aussi.

	Bigrammes	Trigrammes
Nombre de caractéristiques en moyenne	78 375	15 145
Micro-moyenne F-mesure	69,18%	43,55%
Macro-moyenne F-mesure	55,86%	33,20%

TABLE 5.7 – Résultats avec des caractéristiques en n-grammes

On peut voir les premières caractéristiques en bigrammes et trigrammes choisies pour les classes *automotive*, *society* et *technology* (figure 5.3 et 5.4). Les trigrammes deviennent trop spécifiques et aussi peu nombreux à cause du filtrage par le nombre d'occurrences, ce qui reflète la distance des performances entre ces deux tokenisations.

5.6 Affinage du modèle

Le meilleur compromis entre nombre de caractéristiques et performances de la classification a été obtenu avec la spécificité lexicale dont la configuration est rappelée dans le tableau 5.8, sur cette base on a réalisé un affinage du modèle dont l'algorithme est présenté dans la section 4.3. Avec en moyenne 2 500 caractéristiques exclues, pourtant on n'a pas eu une amélioration importante.

candidate;SPECIFICITY	candidate;SPECIFICITY	candidate;SPECIFICITY
radars mobiles;4.900000E-324	laurence rossignol;4.900000E-324	bande passante;4.900000E-324
moteur électrique;4.900000E-324	état végétatif;4.900000E-324	samsung google;4.900000E-324
combi volkswagen;4.900000E-324	autrui gpa;4.900000E-324	lesnouvelles net;4.900000E-324
volkswagen golf;4.900000E-324	comité interministériel;4.900000E-324	the dark;4.900000E-324
dacia logan;4.900000E-324	germaine tillion;4.900000E-324	nouvelle tablette;4.900000E-324
automobiles ccfa;4.900000E-324	prévention routière;4.900000E-324	navigateur chrome;4.900000E-324
wild rubis;4.900000E-324	couples homosexuels;4.900000E-324	bon coin;4.900000E-324
voiture tombe;4.900000E-324	barreaux cnb;4.900000E-324	macbook pro;4.900000E-324
rémy josseaume;4.900000E-324	espace public;4.900000E-324	clash of;4.900000E-324
mm poids;4.900000E-324	van ruymbeke;4.900000E-324	m assange;4.900000E-324
conduite écologique;4.900000E-324	familles monoparentales;4.900000E-324	evolution soccer;4.900000E-324
voiture neuve;4.900000E-324	l'intérieur bernard;4.900000E-324	microsoft sera;4.900000E-324
double embrayage;4.900000E-324	gilles platret;4.900000E-324	acteurs soutenus;4.900000E-324
délégué interministériel;4.900000E-324	mères porteuses;4.900000E-324	jeu officiel;4.900000E-324
prévention routière;4.900000E-324	association prévention;4.900000E-324	internet haut;4.900000E-324
personnes tuées;4.900000E-324	dieter krombach;4.900000E-324	consoles playstation;4.900000E-324
station service;4.900000E-324	sources policières;4.900000E-324	world congress;4.900000E-324
design extérieur;4.900000E-324	secret professionnel;4.900000E-324	utilisateur peut;4.900000E-324
groupe allemand;4.900000E-324	intérieure dgsi;4.900000E-324	dick costolo;4.900000E-324
stéphanie thibault;4.900000E-324	criminalité organisée;4.900000E-324	jeu mobile;4.900000E-324

FIGURE 5.3 – Des caractéristiques en bigrammes choisies pour les classes *automotive*, *society* et *technology*

candidate;SPECIFICITY	candidate;SPECIFICITY	candidate;SPECIFICITY
fiche technique moteur;4.900000E-324	culte musulman cfcm;4.900000E-324	assassin's creed unity;4.900000E-324
boîte double embrayage;4.900000E-324	centres éducatifs fermés;4.900000E-324	google play store;4.900000E-324
moteurs diesel truqués;4.900000E-324	société miss france;4.900000E-324	guitar hero live;4.900000E-324
quatre roues motrices;4.900000E-324	l'organisation etat islamique;4.900000E-324	louis san mis;4.900000E-324
crédits photo daimler;4.900000E-324	paris françois molins;4.900000E-324	the walking dead;4.900000E-324
d'une voiture neuve;4.900000E-324	l' etat islamique;4.900000E-324	the last guardian;4.900000E-324
document cookie split;4.900000E-324	omar ismaïl mostefaï;4.900000E-324	haut débit mobile;4.900000E-324
l'ecologie ségolène royal;4.900000E-324	deuxième langue vivante;4.900000E-324	battle of gods;4.900000E-324
voiture low cost;4.900000E-324	s'est fait exploser;4.900000E-324	sony computer entertainment;4.900000E-324
voitures tour auto;4.900000E-324	actualité aujourd hui;4.900000E-324	street fighter iv;4.900000E-324
mercedes classe e;4.900000E-324	enseignements pratiques interdisciplinaires;4.900000E-324	theft auto v;4.900000E-324
veyron super sport;4.900000E-324	police judiciaire parisienne;4.900000E-324	pro evolution soccer;4.900000E-324
bugatti veyron super;4.900000E-324	l'éducation nationale najat;4.900000E-324	jeu sony disponible;4.900000E-324
automobile club association;4.900000E-324	sécurité intérieure dgsi;4.900000E-324	nvidia shield tv;4.900000E-324
derniers jours combi;4.900000E-324	l'intérieur bernard cazeneuve;4.900000E-324	super smash bros;4.900000E-324
format pave bas;4.900000E-324	gérard depardieu feau;4.900000E-324	jonah lomu rugby;4.900000E-324
split for var;4.900000E-324	médicalement assistée pma;4.900000E-324	premier jeu vidéo;4.900000E-324
radars feux rouges;4.900000E-324	education nationale najat;4.900000E-324	ice cream sandwich;4.900000E-324
renault clio iv;4.900000E-324	depardieu feau immobilier;4.900000E-324	the pirate bay;4.900000E-324
d'un véhicule d'occasion;4.900000E-324	pratiques interdisciplinaires epi;4.900000E-324	massachusetts institute of;4.900000E-324

FIGURE 5.4 – Des caractéristiques en trigrammes choisies pour les classes *automotive*, *society* et *technology*

5.7 Changement de poids de caractéristiques

Pour représenter chaque document dans un vecteur de caractéristiques que le classifieur peut traiter (voir tableau 5.9), on a utilisé le nombre d'occurrences d'une caractéristique dans le document comme son poids, en ignorant toutes les caractéristiques absentes dedans. On n'a pas choisi un indicateur binaire selon que la caractéristique est présente ou pas. Parce que les documents ont une longueur moyenne de 370 mots où il existe forcément une répétitivité des mots.

Tandis que l'écart-type (l'indicateur de dispersion) de la longueur est de 360 mots, soit une grande variation par rapport à la moyenne. Donc on a fait une normalisation du poids en utilisant la valeur du TF-IDF, où TF est la fréquence d'un mot dans un document et IDF la fréquence inverse d'apparition d'un mot dans les documents.

Une expérience a été menée avec la méthode de la spécificité lexicale (seuil =

Méthode (spécificité lexicale, seuil = 0.00001, negativeSpec = false, sans *nbFeatureByTopic*)	modèle initial	modèle affiné
Nombre de caractéristiques en moyenne	68 618	66 112
Micro-moyenne F-mesure	72,14%	72,28%
Macro-moyenne F-mesure	62,46%	62,32%

TABLE 5.8 – Résultats après l'affinage du modèle

	Mot_1	Mot_2	Mot_3	...	Mot_n
Doc_1	3	1	2	...	5

TABLE 5.9 – Représentation d'un document par un vecteur de caractéristiques

0.00001, sans prise en compte de la spécificité négative), et ne garder que les 1000 premières caractéristiques classées par classe. Les résultats sont présentés dans le tableau 5.10.

Avec plus de temps d'exécution, le résultat n'a pas été amélioré, donc on garde le nombre d'occurrences d'une caractéristique dans le document comme son poids.

Poids des caractéristiques	nombre d'occurrences de mot	normalisation par TF*IDF
Micro-moyenne F-mesure	69,86%	62,87%
Macro-moyenne F-mesure	62,02%	53,61%
Temps utilisé	39 minutes	58 minutes

TABLE 5.10 – Comparaison de résultats entre deux poids de caractéristiques utilisés

5.8 Résultats de LIBSVM

Nous avons réalisé une sélection basique de caractéristiques selon les configurations dans le tableau 5.2 (sans aucune méthode de sélection prédéfinie, supprimer les mots dont le nombre d'occurrence est inférieur à 3), puis exécuté le LIBSVM avec la fonction de noyau linéaire et de différentes valeurs du paramètre C. LibSvm nous montre seulement le taux d'exactitude de classification pour chaque partie d'évaluation. Une moyenne de ces dix parties est comparée avec le résultat obtenu par l'algorithme bayésien naïf multinomial : sans aucune méthode de sélection prédéfinie, ainsi qu'avec un filtrage par la spécificité lexicale (seuil du score = 0.00001, sans le seuil *nbFeatureByTopic*).

Dans le tableau 5.11, les résultats de LibSvm reflètent le principe de fonctionnement du paramètre C, expliqué dans la section 3.2. Les performances du modèle sont dégradées quand la valeur du paramètre C est trop grande (trop peu d'erreur

tolérées, ce qui induit un sur-apprentissage) ou trop petite (trop d'erreur tolérées, le modèle est sous-spécifié). Ainsi, la classification par un SVM requiert un ajustement du paramètre de régularisation, même lorsque l'on n'applique pas une méthode de sélection de caractéristiques prédéfinie. Dans tous les cas, elle est beaucoup plus coûteuse en temps.

Après une sélection de caractéristiques par la spécificité lexicale, le résultat baisse légèrement à 73,56%. En nous rappelant que ce résultat est atteint avec une réduction de 57% de caractéristiques et une réduction importante de temps d'exécution, appliquer une sélection de caractéristiques efficace avant de construire les modèles par l'algorithme bayésien naïf multinomial convient bien à notre contexte de travail.

	LIBSVM, sans application de méthode de sélection	Bayésien Naïf Multinomial, sans application de méthode de sélection	Bayésien Naïf Multinomial + filtrage par la spécificité lexicale
Exactitude moyenne	C = 0,001, 70,38% C = 0,01, 73,93% C = 0,1, 73,25% C = 1, 72,11% C = 10, 71,23%	74,06%	73,56%

TABLE 5.11 – Résultats de LIBSVM

CHAPITRE 6

DISCUSSION

Dans la plupart des littératures citées, les caractéristiques sont choisies globalement. Dans notre cas, nous avons gardé les scores des caractéristiques par rapport à chaque classe, puis les filtrer par les seuils. La raison de notre choix est que la méthode de la spécificité lexicale a été implémentée en amont de ce travail, et on veut comparer les trois autres méthodes avec elle selon ce même algorithme. C'est également pour avoir une meilleure interprétation des caractéristiques choisies pour chaque catégorie afin de pouvoir contribuer plus au futur travail.

Dans nos différentes expériences, le CPD a obtenu des résultats légèrement inférieurs à ceux de la spécificité lexicale avec une réduction importante du nombre de caractéristiques. On a gardé des caractéristiques dont la valeur absolue de score est supérieure ou égale à son meilleur seuil (0.99). Puis quand il existe une valeur du paramètre *nbFeatureByTopic* qui garde les n caractéristiques les mieux classées pour chaque classe, on trie les scores dans l'ordre descendant, soit privilégier les scores positifs. Ainsi les caractéristiques sélectionnées ont le score 0.99, 1.0, -0.99 ou -1.0 (valeur approximative, car on ignore le score -1.0 qui indique que la caractéristique n'est pas présente dans la classe). Les scores positifs signifient que ces caractéristiques figurent (presque) seulement dans la classe courante, et ceux négatifs indiquent qu'ils ne figurent presque pas dans cette classe.

Selon les découvertes dans [Yang and Pedersen, 1997], l'information mutuelle (MI) obtient de faibles performances à cause de son biais de préférence pour les termes rares et sa sensibilité aux erreurs d'estimation de probabilité. Dans nos expériences, l'information mutuelle (plus précisément *Pointwise Mutual Information*, voir section 2.3), avec son meilleur seuil 1.0, il atteint des résultats similaires avec les autres méthodes mais en utilisant beaucoup plus de caractéristiques. De ce fait, nous lui avons augmenté le seuil du nombre d'occurrences minimal à 10, ce qui l'aide à avoir une meilleure macro-moyenne F-mesure en utilisant une moitié moins de caractéristiques (voir tableau 6.1).

Quand la spécificité négative est aussi prise en compte, on rajoute les caractéristiques sous-représentées dont la valeur absolue du score est inférieure au seuil, qui sont généralement moins nombreuses que les caractéristiques sur-représentées. Les résultats présentés dans le tableau 6.2 montrent que les performances restent similaires.

Concernant le TF-IDF, comme ce qu'on a vu dans la figure 5.1, si on veut avoir une meilleure macro-moyenne F-mesure, il vaut mieux baisser le nombre de caractéristiques à retenir par classe. Par exemple si on ne retient que les 3000 premières caractéristiques par classe, qui donne 53 917 caractéristiques en total, on obtiendra

Méthode (Information mutuelle, seuil = 1.0)	minOcc = 3	minOcc = 10
Nombre de caractéristiques en moyenne	157 267	81 155
Micro-moyenne F-mesure	72,43%	72,18%
Macro-moyenne F-mesure	59,72%	62,30%
F-mesure sur des classes difficiles (*careers, datingLoveCouple, outing*)	1,94%, 20,31%, 22,70%	17,63%, 45,11%, 33,75%

TABLE 6.1 – Comment améliorer la méthode de l'information mutuelle

Méthode (spécificité lexicale, seuil = 0.00001, sans *nbFeatureByTopic*)	negativeSpec = false	negativeSpec = true
Nombre de caractéristiques en moyenne	68 618	69 342
Micro-moyenne F-mesure	72,14%	72,19%
Macro-moyenne F-mesure	62,46%	62,43%
F-mesure sur des classes difficiles (*careers, datingLoveCouple, outing*)	19,43%, 44,85%, 33,40%	17,95%, 43,21%, 34,14%

TABLE 6.2 – Performances de la spécificité lexicale quand les spécificités négatives sont prises en compte

une macro-moyenne F-mesure de 62.07%, au lieu de 59.17% par 156 404 caractéristiques. Ceci montre que le TF-IDF a des difficultés à filtrer correctement le bruit.

Regardons les vingt premières caractéristiques sélectionnées pour la classe *automotive* et *careers* (voir figure 6.1 et 6.2), la plupart des caractéristiques choisies par la spécificité lexicale sont étroitement liés à l'automobile. Dans la liste du CPD et de l'information mutuelle, il existe des mots moins spécifiques à l'automobile que ceux choisis par la spécificité lexicale, par exemple « jovanka » et « barbanti ». Et le TF-IDF a choisi encore plus de caractéristiques loin du sujet.

La classe *careers* qui détient 396 documents est le moins représentant parmi les 27 classes, qui cause souvent un point de chute. Les caractéristiques choisies pour cette classe se ressemblent pour les quatre méthodes : des noms de sites de soutien scolaire, d'annonce d'entraide quotidienne et de recrutement, ou des noms de personne, etc.

La figure 6.3 montre les mesures de précision, rappel et F-mesure par classe. La catégorie *sports* contient le plus de documents et reçoit la meilleur classification. La classe *careers* obtient le plus bas rappel et F-mesure comme ce qu'on a mentionné au-dessus. Par contre, peu de documents ne conduit pas toujours à une mauvaise performance et *vice versa*. Par exemple même si la classe *society* contient 6 716 docu-

candidate;SPECIFICITY	candidate;CPD	candidate;TF_IDF	candidate;MUTUAL_INFORMATION
parigné;4.900000E-324	hypersport;1.00	crtg;3.476802E-03	ncap;7.661831E+00
alltrack;4.900000E-324	linian;1.00	seniorplanet;1.738401E-03	zaniroli;7.661831E+00
gle;4.900000E-324	ncap;1.00	smartads;1.738401E-03	picanto;7.661831E+00
inrix;4.900000E-324	zaniroli;1.00	formatid;1.738401E-03	cyl;7.661831E+00
acda;4.900000E-324	vendezvotrevoiture;1.00	pageid;1.570073E-03	fémininscritérespeut;7.661831E+00
chevrolet;4.900000E-324	êvêque;1.00	split;1.158934E-03	hélica;7.661831E+00
talisman;4.900000E-324	picanto;1.00	cookie;8.400304E-04	bodrogi;7.661831E+00
stationnements;4.900000E-324	cyl;1.00	undefined;7.477957E-04	motobot;7.661831E+00
méhari;4.900000E-324	kidioui;1.00	typeof;7.477957E-04	fcv;7.661831E+00
uppsala;4.900000E-324	niedereimer;1.00	sponso;7.041537E-04	airbumps;7.661831E+00
logan;4.900000E-324	boxster;1.00	name;5.794671E-04	barbanti;7.661831E+00
berlines;4.900000E-324	verbalisateur;1.00	pave;5.760806E-04	lambroghini;7.661831E+00
i;4.900000E-324	tnga;1.00	var;5.353871E-04	fxs;7.661831E+00
abs;4.900000E-324	fémininscritèrespeut;1.00	sas;5.049859E-04	laliron;7.661831E+00
capote;4.900000E-324	hélica;1.00	if;4.288920E-04	tricorps;7.661831E+00
carrera;4.900000E-324	nadjowski;1.00	target;4.288920E-04	lykan;7.661831E+00
tfsi;4.900000E-324	jovanka;1.00	planet;2.669278E-04	xje;7.661831E+00
fortwo;4.900000E-324	mwt;1.00	function;2.291524E-04	radardroid;7.661831E+00
contraventions;4.900000E-324	motospot;1.00	radars;2.002702E-04	airbump;7.661831E+00

FIGURE 6.1 – Les meilleures caractéristiques classées pour la classe *automotive*

candidate;SPECIFICITY	candidate;CPD	candidate;TF_IDF	candidate;MUTUAL_INFORMATION
chegg;4.900000E-324	chegg;1.00	officeteam;1.420020E-04	officeteam;8.420276E+00
boya;4.900000E-324	officeteam;1.00	aliza;1.336490E-04	codingame;8.420276E+00
contractuels;4.900000E-324	jissey;1.00	jabès;1.336490E-04	petithuguenin;8.420276E+00
taskrabbit;4.900000E-324	codingame;1.00	employeur;1.139959E-04	gabs;8.420276E+00
zawieja;4.900000E-324	gaignard;1.00	combrexelle;9.195236E-05	chronopsychologues;8.420276E+00
giphy;4.900000E-324	sounigo;1.00	salariés;9.164337E-05	milléniaux;8.420276E+00
psychosociaux;4.900000E-324	psya;1.00	alumni;8.909931E-05	bréhin;8.420276E+00
duhoux;4.900000E-324	gabs;1.00	recrutement;8.152810E-05	sihad;8.420276E+00
jabès;4.900000E-324	buzzfed;1.00	salarié;7.629839E-05	disruptive;8.420276E+00
aliza;4.900000E-324	chronopsychologues;1.00	bni;7.517755E-05	chegg;8.420276E+00
assouplissements;4.900000E-324	adary;1.00	entreprises;7.431489E-05	jissey;8.420276E+00
officeteam;4.900000E-324	soes;1.00	burn;7.039050E-05	sounigo;8.420276E+00
origgi;4.900000E-324	milléniaux;1.00	apec;6.796777E-05	psya;8.420276E+00
apec;4.900000E-324	assaël;1.00	recruteur;6.782306E-05	origgi;8.420276E+00
duez;4.900000E-324	rfs;1.00	chegg;6.682449E-05	buzzfed;8.420276E+00
travail;4.900000E-324	bréhin;1.00	utt;6.596329E-05	soes;8.420276E+00
malakoff;4.900000E-324	letsbuyit;1.00	duhoux;6.596329E-05	rfs;8.420276E+00
graphologie;4.900000E-324	altermondes;1.00	linkedin;6.483336E-05	letsbuyit;8.420276E+00
fessiers;4.900000E-324	sihad;1.00	viadeo;6.474019E-05	altermondes;8.420276E+00

FIGURE 6.2 – Les meilleures caractéristiques classées pour la classe *careers*

ments, la F-mesure atteint seulement 32,79%. En revanche, *weddings* et *automotive* qui sont composés respectivement de 880 et 682 documents, obtiennent la F-mesure de 70,95% et 72,06%. Ceci montre l'importance de la qualité des données d'apprentissage des classes délicates, et c'est aussi le problème auquel on se confronte en ce moment.

On peut analyser les erreurs de classification grâce à la matrice de confusion (voir figure **??**), où les lignes représentent les classes de référence, et les colonnes représentent les classes de prédiction. En conséquence, les nombres sur la diagonale montrent la quantité de documents correctement classifiés pour chaque classe, et toutes les autres valeurs signifient le nombre de documents faussement classifiés. Par exemple en haut à gauche, la table nous montre qu'il y a 380 documents annotés avec la classe *culture_entertainment* mais qui sont classifiés dans la classe *business*.

Pour chaque colonne de prédiction, nous avons mis en évidence les trois premières erreurs de classification les plus nombreuses. Parmi ces erreurs, certaines sont moins graves car les classes sont sémantiquement liées, par exemple *culture_entertainment* et *outing*. Mais certaines erreurs ne sont pas négligeables : 58 *business* sont classés

1	Category name	Expected	TruePositive	FalsePositive	Precision	Recall	F-Mesure
2	xiko:topic.careers	396	55	115	32.35 %	13.89 %	19.43 %
3	xiko:topic.society	6716	2042	3698	35.57 %	30.41 %	32.79 %
4	xiko:topic.outing	691	243	521	31.81 %	35.17 %	33.40 %
5	xiko:topic.legal_issue	1785	903	1802	33.38 %	50.59 %	40.22 %
6	xiko:topic.dating_love_couple	474	231	325	41.55 %	48.73 %	44.85 %
7	xiko:topic.news	6456	4157	4717	46.84 %	64.39 %	54.23 %
8	xiko:topic.environment	4049	2375	1773	57.26 %	58.66 %	57.95 %
9	xiko:topic.family_parenting	2233	1165	615	65.45 %	52.17 %	58.06 %
10	xiko:topic.business	7814	4993	3949	55.84 %	63.90 %	59.60 %
11	xiko:topic.spirituality	574	302	131	69.75 %	52.61 %	59.98 %
12	xiko:topic.style_fashion	2833	1892	1367	58.05 %	66.78 %	62.11 %
13	xiko:topic.education	1260	977	873	52.81 %	77.54 %	62.83 %
14	xiko:topic.science	5158	2902	980	74.76 %	56.26 %	64.20 %
15	xiko:topic.health_well_being	9931	5571	1541	78.33 %	56.10 %	65.38 %
16	xiko:topic.beauty	1787	1192	627	65.53 %	66.70 %	66.11 %
17	xiko:topic.pets	2020	1578	1154	57.76 %	78.12 %	66.41 %
18	xiko:topic.real_estate	1698	1146	567	66.90 %	67.49 %	67.19 %
19	xiko:topic.travel	3332	2548	1578	61.75 %	76.47 %	68.33 %
20	xiko:topic.weddings	880	530	84	86.32 %	60.23 %	70.95 %
21	xiko:topic.automotive	682	543	282	65.82 %	79.62 %	72.06 %
22	xiko:topic.personal_finance	3096	2449	1073	69.53 %	79.10 %	74.01 %
23	xiko:topic.technology	6747	5095	1681	75.19 %	75.52 %	75.35 %
24	xiko:topic.politics	7666	5891	1852	76.08 %	76.85 %	76.46 %
25	xiko:topic.culture_entertainment	20513	15842	4083	79.51 %	77.23 %	78.35 %
26	xiko:topic.cooking	2876	2481	761	76.53 %	86.27 %	81.10 %
27	xiko:topic.home_garden	2726	2227	448	83.25 %	81.69 %	82.47 %
28	xiko:topic.sports	32428	29374	1520	95.08 %	90.58 %	92.78 %

FIGURE 6.3 – Tableau de résultats par classe agrégés des dix plis, avec en moyenne 68 618 caractéristiques sélectionnés par la spécificité lexicale (seuil = 0.00001, negativeSpec=false)

en *cooking*, 382 *health_well_being* sont classés en *business*, ou encore 265 *sports* sont classés en *legal_issue*. La classe *society* est souvent mélangée avec d'autres sujets, ce qui est confirmé par de nombreuses valeurs non sur la diagonale et ceci apporte un rappel et une précision tous bas. La distribution des classes (voir tableau 4.1) nous rappelle qu'il existe quand même une sur-représentation de certaines classes, donc dans des futurs travaux, avec l'aide de la matrice de confusion, on peut réajuster le corpus pour baisser cet impact.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB
1		business	culture_entertainment	home_garden	style_fashion	news	health_well_being	travel	automotive	beauty	environment	technology	careers	science	weddings	family_parenting	education	politics	real_estate	outing	pets	society	spirituality	dating_love_couple	legal_issue	personal_finance	sports	cooking
2	business	4993	230	11	53	327	79	57	34	9	207	378	69	23		4	139	406	107	25	22	224	3		50	264	42	58
3	culture_entertainment	380	15842	57	355	533	112	448	6	66	150	403	1	87	4	57	46	352	47	166	69	581	7	8	175	5	393	163
4	home_garden	18	24	2227	92		12	73	2	10	53	26		22	11	20	1	1	72	1	6	2		1	4	27	3	18
5	style_fashion	213	314	43	1892	47	9	69	1	46	16	20		3	17	16	3	13	25		5	24		4	3	7	21	22
6	news	137	164	3	8	4157	24	17	8		221	33	1	9		2	13	111	10	130	37	442	3		370		544	12
7	health_well_being	382	591	52	295	156	5571	61	2	315	157	91	3	236	23	401	36	57	6	6	135	541	34	287	89	14	158	232
8	travel	43	222	15	22	4	6	2548	39	4	28	69		32		4	8	4	20	62	12	21			1	133	16	19
9	automotive	51	4		3			1	543		7	9		1					3			3			4	47	6	
10	beauty	6	93	8	129	2	192	17		1192	10	13		7	11	3		2		1		13					39	49
11	environment	218	70	55	11	326	62	83	19	12	2375	40	3	199		1	7	108	18	4	264	119	3		20	1	7	24
12	technology	362	204	8	13	103	29	21	77	3	43	5095	2	212		5	20	44	12	3	19	210	1		31	206	23	1
13	careers	141	14			3	20	2		3		12	55	1		10	90	7	1			26		4	1	3	3	
14	science	105	83	36	3	112	474	43		7	466	171	2	2902		7	65	19	3		522	97			11	2	7	21
15	weddings		54	29	137	2	1	18		72				1	530	7		2		1		3	5	3		1	6	8
16	family_parenting	5	331	64	138	6	119	35	2	51	4	41		4	17	1165	34	1	1	3	11	12	12		1	22	110	44
17	education	21	29	1	1	4	13	6	1		4	16	2	4		20	977	16	7	2	2	127	1	1		3	1	1
18	politics	532	114		2	239	15	5			59	23		3			32	5891	9	4	1	522	2		184	15	13	1
19	real_estate	46	22	32	3	2		109			22	12		4			4	7	1146			22			4	262	1	
20	outing	6	200	3	11	17		100	9		1	4	1			1	3		1	243	1	2	1		4	6	28	49
21	pets	31	25	1	4	161	20	3			74	6		67		1	3	1	1	1	1578	33			3		4	3
22	society	523	636	1	24	1089	208	88	33	10	165	129	29	34		29	326	511	43	7	33	2042	58	6	571	26	66	29
23	spirituality	1	93	3	3	17	3	5			3	1		1		2	3	4	1	4		110	302	9	7		2	
24	dating_love_couple	1	148	3	13	1	22	7		5		10			1	12	1				1			231	4	1	11	2
25	legal_issue	66	61	2	2	224	10	29	2		9	13		2		6	14	99	8		3	297	1	1	903	19	11	3
26	personal_finance	287	8		2	1	39	15	10		11	80		14			3	8	162			4				2449	3	
27	sports	305	303	3	34	1256	53	214	29	9	47	76	2	11		2	20	77	8	91	6	240			265	1	29374	2
28	cooking	69	46	18	9	85	19	52	8	5	16	5		3		5	2	2	2	10	5	23		1		8	2	2481

FIGURE 6.4 – Table de confusion par classe (même configuration que celle de la figure 6.3)

CONCLUSION ET PERSPECTIVES

Conclusion

Dans ce travail nous avons évalué quatre méthodes de sélection de caractéristiques en vue d'améliorer la rapidité et l'exactitude de notre classifieur de thématiques. A la différence de beaucoup de travaux précédents, notre corpus possède ses propres particularités et les caractéristiques ont été sélectionnées par rapport à chaque classe. De plus, nous avons réalisé une validation croisée qui intègre la sélection de caractéristiques sur chaque sous-échantillon d'apprentissage, afin d'éviter l'effet du sur-apprentissage. L'évaluation par l'algorithme Bayésien Naïf Multinomial montre que la spécificité lexicale est la meilleure méthode, qui ne dégrade pas les résultats en réduisant 57% de caractéristiques, et elle apporte également une meilleure macro-moyenne F-mesure. L'application des différents prétraitements (racinisation, lemmatisation, tokenisation en n-grammes) n'apporte pas de gain, voire baisse les performances. Sans aucune méthode de sélection, le taux d'exactitude en moyenne obtenu par les SVM (73.93%), avec un paramètre de régularisation (C = 0.01), est légèrement supérieur à celui obtenu après la sélection par la spécificité lexicale et l'apprentissage par le Bayésien Naïf Multinomial (73.56%).

Perspectives

Les futurs travaux prévoient de mettre en place d'autres méthodes de sélection qui obtiennent de bonnes performances dans la littérature, par exemple le gain d'information et le CHI-deux. On peut aussi envisager d'implémenter l'algorithme qui choisit les caractéristiques globalement au lieu de les choisir par classe. D'autres algorithmes d'apprentissage restent à essayer, par exemple les réseaux de neurones, la régression logistique, etc. A l'égard de l'approche de l'apprentissage non supervisé, ses avantages sont de ne plus s'appuyer sur des mots eux-mêmes figurant dans le corpus et l'indépendance sur un corpus annoté pourraient aider à pallier notre présent problème.

BIBLIOGRAPHIE

[AGRAWAL et al., 1995] AGRAWAL, R., Heikki, M., Ramakrishnan, S., Toivonen, H., and A. Inkeri, V. (1995). Fast discovery of association rules. In *Advances in Knowledge Discovery and Data Mining*, pages 307—-328. – Cité page 31.

[Aït-Hamlat, 2016] Aït-Hamlat, J. (2016). Apprentissage bayésien incrémental pour la détermination de l'âge et du genre d'utilisateurs de plateformes du web social. In *Actes de la conférence conjointe JEP-TALN-RECITAL 2016*, volume 3. – Cité page 31.

[Bayes, 1763] Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. of the Royal Soc. of London*, 53:370–418. – Cité page 23.

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. – Cité page 26.

[Drucker et al., 1999] Drucker, H., Wu, D., and Vapnik, V. N. (1999). Support vector machines for spam categorization. *Trans. Neur. Netw.*, 10(5):1048–1054. – Cité page 11.

[Dumais et al., 1998] Dumais, S., Platt, J., Heckerman, D., and Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 148–155, New York, NY, USA. ACM. – Cité page 11.

[Eensoo et al., 2015] Eensoo, E., Nouvel, D., Martin, A., and Valette, M. (2015). Combiner analyses textométriques, apprentissage supervisé et représentation vectorielle pour l'analyse de la subjectivité. In *22 ème Traitement Automatique des Langues Naturelles*. – Cité pages 14 et 17.

[Forman, 2003] Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *J. Mach. Learn. Res.*, 3:1289–1305. – Cité pages 14 et 34.

[Forman, 2007] Forman, G. (2007). Feature selection for text classification. In Huan, L. and Hiroshi, M., editors, *Computational Methods of Feature Selection (Chapman & Hall / Crc Data Mining and Knowledge Discovery Series)*, chapter 13, pages 257–276. Chapman & Hall/CRC. – Cité page 11.

[Fürnkranz, 1998] Fürnkranz, J. (1998). A study using n-gram features for text categorization. – Cité page 31.

[Hall et al., 2009] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18. – Cité pages 23 et 30.

[Ifrim et al., 2008] Ifrim, G., Bakir, G., and Weikum, G. (2008). Fast logistic regression for text categorization with variable-length n-grams. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 354–362, New York, NY, USA. ACM. – Cité page 11.

[Jing et al., 2002] Jing, L.-P., Huang, H.-K., and Shi, H.-B. (2002). Improved feature selection approach tf-idf in text mining. In *Proceedings of the First International Conference on Machine Learning and Cybernetics*. – Cité page 18.

[Joachims, 1998] Joachims, T. (1998). Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK. Springer-Verlag. – Cité pages 11, 13 et 25.

[Jurafsky and Martin, 2015] Jurafsky, D. and Martin, J. H., editors (2015). *Speech and Language Processing*. – Cité pages 13 et 33.

[Klinkenberg and Joachims, 2000] Klinkenberg, R. and Joachims, T. (2000). Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 487–494, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. – Cité page 11.

[Labbe and Labbe, 1997] Labbe, C. and Labbe, D. (1997). Que mesure la spécificité du vocabulaire ? – Cité page 17.

[Lafon, 1980] Lafon, P. (1980). Sur la variabilité de la fréquence des formes dans un corpus. *Mots*, 1(1):127–165. – Cité page 17.

[Larkey and Croft, 1996] Larkey, L. S. and Croft, W. B. (1996). Combining classifiers in text categorization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 289–297, New York, NY, USA. ACM. – Cité page 11.

[Lewis and Ringuette, 1994] Lewis, D. D. and Ringuette, M. (1994). A comparison of two learning algorithms for text categorization. In *In Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93. – Cité page 11.

[Lilleberg et al., 2015] Lilleberg, J., Zhu, Y., and Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In *14th IEEE International Conference on Cognitive Informatics & Cognitive Computing, ICCI*CC 2015, Beijing, China, July 6-8, 2015*, pages 136–140. – Cité page 14.

[Lin et al., 2015] Lin, Y., Lei, H., Wu, J., and Li, X. (2015). An empirical study on sentiment classification of chinese review using word embedding. *CoRR*, abs/1511.01665. – Cité page 14.

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781. – Cité page 14.

[Mosteller and Lee Wallace, 1964] Mosteller, F. and Lee Wallace, D. (1964). *Inference and Disputed Authorship : The Federalist*. – Cité page 23.

[Newsgroups, 1999] Newsgroups (1999). http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html. – Cité page 20.

[Ohsumed, 2005] Ohsumed (2005). http://davis.wpi.edu/~xmdv/datasets/ohsumed.html. – Cité page 13.

[Porter, 2001] Porter, M. F. (2001). Snowball: A language for stemming algorithms. – Cité pages 30 et 36.

[Reuters-21578, 1997] Reuters-21578 (1997). http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. – Cité page 13.

[Salem et al., 2003] Salem, A., Lamalle, C., Martinez, W., Fleury, S., Fracchiolla, B., Kuncova, A., and Maisondieu, A. (2003). *Lexico3 Outils de statistique textuelle. Manuel d'utilisation*. Syled-CLA2T, Université Sorbonne Nouvelle. – Cité page 17.

[Salton and Buckley, 1988] Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523. – Cité page 18.

[Schmid, 1994] Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*. – Cité page 30.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Comput. Surv.*, 34(1):1–47. – Cité pages 13 et 19.

[Simeon and Hilderman, 2008] Simeon, M. and Hilderman, R. (2008). Categorical proportional difference: A feature selection method for text categorization. In *Proceedings of the 7th Australasian Data Mining Conference - Volume 87*, AusDM '08, pages 201–208, Darlinghurst, Australia, Australia. Australian Computer Society, Inc. – Cité pages 20 et 21.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA. – Cité page 25.

[Wang and Manning, 2012] Wang, S. and Manning, C. D. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 90–94, Stroudsburg, PA, USA. Association for Computational Linguistics. – Cité page 14.

[Wiener and Weigend, 1995] Wiener, E. D., P. J. O. and Weigend, A. S. (1995). A neural network approach to topic spotting. In *Proceedings of SDAIR-95, 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 317–332. – Cité page 11.

[Xu et al., 2007] Xu, Y., Jones, G., Li, J., Wang, B., and Sun, C. (2007). A study on mutual information-based feature selection for text categorization. *Journal of Computational Information Systems*, pages 1007–1012. – Cité page 20.

[Yang and Liu, 1999] Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49, New York, NY, USA. ACM. – Cité page 25.

[Yang and Pedersen, 1997] Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. pages 412–420. Morgan Kaufmann Publishers. – Cité pages 11, 13, 14, 19 et 43.

ANNEXE A

EXTRAIT DE CORPUS

Le corpus est un fichier CSV qui contient trois colonnes : la catégorie de référence, l'URL de XiKO et le contenu à classifier, tel que montre un extrait ci-dessous :

xiko : topic.real_estate,http : //www.xiko.fr,"Exemple de newsletter. Vous aimez nos articles ? Recevez-les en premier !"

xiko : topic.real_estate,http : //www.xiko.fr,"Question de Noémie (Bayonne) : Mon mari et moi avons signé, il y a 15 jours un compromis de vente pour l'achat d'un appartement. Entre-temps, nous avons trouvé un logement qui nous convient mieux. Peut-on encore se rétracter ? Un compromis équivaut à une vente. Il s'agit d'un contrat qui engage le vendeur et l'acheteur à vendre un bien pour un prix donné. La signature engage les deux parties et permet de reporter la vente effective au jour de la signature de l'acte authentique de vente auprès du notaire. A priori, vous ne pouvez changer d'avis. Seule la non-réalisation de l'une des conditions suspensives prévues dans le compromis de vente peut permettre de se désengager. Il s'agit le plus souvent de conditionner la vente à l'octroi du financement adéquat, à un accord de la mairie pour réaliser d'éventuels travaux, à un certificat de non-hypothèque du logement, à l'absence de droit de préemption d'un éventuel locataire, ou bien encore à une servitude qui déprécie la valeur du bien. Or dans votre cas, aucun de ces motifs ne peut semble-t-il être mis en avant. Il vous reste toutefois une ultime chance pour vous tirer d'affaires. Si votre compromis de vente prévoit que l'acheteur ou le vendeur peut se rétracter sans motif, alors vous pourrez faire machine arrière. En terme juridique, on parle de clause de dédit. Celle-ci prévoit que vous puissiez renoncer à l'acquisition. En contrepartie, vous devrez abandonner au vendeur l'acompte que vous avez versé à la signature du compromis de vente. Une somme qui peut aller jusqu'à 10% de la valeur du bien. Exemple de newsletter. Vous aimez nos articles ? Recevez-les en premier !"

xiko : topic.culture_entertainment-television,http : //www.xiko.fr,"(Relaxnews) - Garou, Florent Pagny, Mika et Zazie sont les héros d'un petit film mis en ligne par TF1 annonçant la prochaine saison de The Voice. Tourné dans les célèbres studios de la Plaine-Saint-Denis, les 4 coachs y font preuve de malice et de beaucoup d'adresse pour venir bidouiller leurs sièges et ceux de leurs rivaux afin d'être sûr de pouvoir faire équipe avec les meilleurs jeunes talents qui se présenteront aux auditions. googletag.cmd.push(function() { googletag.display('div-gpt–732377866'); }); La date de diffusion de la première émission, toujours présentée par Nikos Aliagas, devrait être annoncée début janvier 2016. Avec le départ de Jenifer, Florent Pagny est désormais le seul coach à avoir participé à toutes les saisons de The Voice."

ANNEXE B

EXTRAIT DE CODES

Voici un extrait de la classe principale *TrainTopicCommand* :

```
public final class TrainTopicCommand {

    private static final Logger LOGGER = LoggerFactory.getLogger(TrainTopicCommand.class);
    public static final int PROPERTIES_FILE_ARG_POS = 0;

    public static void main(final String[] args) throws Exception {

        DateTime start = new DateTime();

        CommandLine commandLine = TopicTrainArguments.buildCommandLine(args);

        TopicTrainArguments arguments = new TopicTrainArguments(args[PROPERTIES_FILE_ARG_POS]);
        arguments.getOutputDirPath().toFile().mkdirs();
        PropertiesFileWriter.write(arguments);

        LOGGER.info("Load train set");
        TrainSetMaker trainSetMaker = new TrainSetMaker(arguments);
        trainSetMaker.process(Files.newInputStream(arguments.trainSetPath()));
        TrainSet trainSet = trainSetMaker.getTrainSet();
        TrainSetDispatcher trainSetDispatcher = trainSetMaker.getTrainSetDispatcher();

        LOGGER.info("Select and write features");
        FeatureDirectoryWriter featureDirectoryWriter = new FeatureDirectoryWriter(arguments);
        featureDirectoryWriter.selectAndWriteFeatures(trainSet);
        FeatureDirectoryReader featureDirectoryReader = new FeatureDirectoryReader(arguments);

        Set<TextFeatureCandidate> selectedFeatures ;
        if (arguments.NbLocalFeatureSelectors() == 0){
            selectedFeatures = featureDirectoryReader.readGlobalFeatures();
        } else {
            selectedFeatures = featureDirectoryReader.readLocalFeatures();
        }

        LOGGER.info("Build Model and Evaluate for the total corpus");
        buildWekaModelAndEvaluate(arguments, trainSet, selectedFeatures, "initial");

        if (arguments.isExcludedFeatureActive()) {

            LOGGER.info("{EXCLUDE FEATURE} Choose top features causing error for the total corpus");
            ExcludedFeaturesGenerator excludedFeaturesGenerator = new
                ExcludedFeaturesGenerator(arguments, "initial");
            Set<TextFeatureCandidate> excludedFeatures = excludedFeaturesGenerator.
                excludedFeatures();
            excludedFeaturesGenerator.writeExcludedFeatures("excludedFeatures.csv");
            Set<TextFeatureCandidate> updatedFeatures = Sets.difference(selectedFeatures,
                excludedFeatures);

            LOGGER.info("{EXCLUDE FEATURE} Build Model and Evaluate");
            buildWekaModelAndEvaluate(arguments, trainSet, updatedFeatures, "withExcludedFeatures")
                ;
```

```
        }

        if (arguments.isCrossFeatureSelectionActive(commandLine)) {
            LOGGER.info("Dispatch the corpus into ten evaluation sets");
            EvaluationTrainSetMaker evalTrainSetMaker = new EvaluationTrainSetMaker(arguments,
                trainSetDispatcher);
            evalTrainSetMaker.process(Files.newInputStream(arguments.trainSetPath()));
            List<TrainSet> evaluationTrainSets = evalTrainSetMaker.getEvaluationTrainSets();

            LOGGER.info("Select and write features for ten train sets");
            CrossFeatureDirectoryWriter crossFeatureDirWriter = new
                CrossFeatureDirectoryWriter(arguments);
            crossFeatureDirWriter.selectAndWriteFeatures(trainSet, evaluationTrainSets);
            CrossFeatureDirectoryReader crossFeatureDirectoryReader = new
                CrossFeatureDirectoryReader(arguments);
            List<Set<TextFeatureCandidate>> crossSelectedFeatures =
                crossFeatureDirectoryReader.readCrossValidationFeatures();

            LOGGER.info("Build Model and Evaluate for ten train sets");
            buildCrossWekaModelAndEvaluate(arguments, trainSet, trainSetDispatcher,
                evaluationTrainSets, crossSelectedFeatures, "initial_cross");

            if (arguments.isExcludedFeatureActive()) {
                LOGGER.info("{EXCLUDE FEATURE} Choose top features causing error for ten sets");
                List<Set<TextFeatureCandidate>> updatedCrossSelectedFeatures = new ArrayList
                    <>();

                for (int i = 0; i < 10; i++) {
                    ExcludedFeaturesGenerator crossExcludedFeaturesGenerator = new
                        ExcludedFeaturesGenerator(arguments, "initial_cross", "" + i);
                    Set<TextFeatureCandidate> crossExcludedFeatures =
                        crossExcludedFeaturesGenerator.excludedFeatures();
                    crossExcludedFeaturesGenerator.writeExcludedFeatures("excludedFeatures" + i + ".
                        csv");
                    updatedCrossSelectedFeatures.add(Sets.difference(crossSelectedFeatures.get(i)
                        , crossExcludedFeatures));
                }

                LOGGER.info("{EXCLUDE FEATURE} Build Model and Evaluate");
                buildCrossWekaModelAndEvaluate(arguments, trainSet, trainSetDispatcher,
                    evaluationTrainSets, updatedCrossSelectedFeatures, "withExcludedFeatures_cross"
                    );
            }
        }

        DateTime end = new DateTime();
        arguments.fetchResultWriter().write((end.getMillis() - start.getMillis()) / 60000 + " total
            minutes used" + "\n");

        arguments.fetchResultWriter().close();
    }
  }
}
```