Prédire le parti politique québécois à partir de discours

Louis-Émile Robitaille
Université Laval
louis-emile.robitaille.1@ulaval.ca

Martin Richard
Université Laval
martin.richard-cerda.1@ulaval.ca

Arnaud Cavrois
Université Laval
arnaud.cavrois.1@ulaval.ca

23 décembre 2016

Résumé

Dans une campagne électorale, l'analyste politique étudie les actions des candidats, leurs discours et le contexte social pour faire des liens et donner un sens au déroulement de la campagne. Une partie de leur travail revient à faire du classement : placer les actions d'un candidat dans une certaine idéologie ou tendance politique et y associer un discours, une vision ou un mouvement social. Le présent article vise à vérifier si des algorithmes d'apprentissage peuvent réussir à classer efficacement des discours en jouant sur la structure représentant ceux-ci.

I. Présentation du problème

Dans une ère où l'information est disponible en très grande quantité et où elle se répand rapidement, le besoin de vérifier la qualité d'un article devient de plus en plus nécessaire. Identifier l'origine d'un texte est certainement le meilleur moyen d'assurer un début de qualité dans ce que l'on consulte comme information. Dans le cas des discours politiques, le problème qui se pose est de trouver un moyen automatisé de les classer selon leur appartenance politique.

II. État de l'art

En cherchant judicieusement, il est possible de trouver plusieurs articles qui traitent d'apprentissage en analyse de textes. Plus précisément, l'article de [1] révèle qu'il est possible de reconnaître certaines caractéristiques d'un texte comme la subjectivé et le neutralité. En effet, en utilisant une représentation des données par phrase (contrairement aux bag-of-words), les auteurs montrent qu'un classifieur bayésien naïf peut prédire la subjectivité d'un texte avec une probabilité allant jusqu'à 64.87%. Dans le cas d'un classifieur de type `entropie maximale`, ils montrent que l'algorithme peut prédire la nature subjective d'un texte avec une probabilité allant jusqu'à 67.73%. De plus, en regroupant les phrases en séquences de phrases, l'article nous apprend que les taux de succès peuvent grimper jusqu'à 77.42%.

Un autre article notable [2] présente une méthode d'analyse de la polarité d'un message (type twitter), son degré de positivité ou de négativité. Celle-ci peut-être utilisée comme base efficace dans la discrimination de textes courts. L'algorithme utilise une approche multilinguiste se basant sur des familles de langages comme les langues Germaniques ou Romanes. Pour cela, il travaille préalablement les données en prenant compte des abrévia-

tions, des émoticônes, des stop words, des racines, des n-grams et des q-grams pour ensuite former une représentation du texte. Les textes sont alors sous la forme de sacs de mots transformés par TF-IDF (Term Frenquency -Inverse Document Frenquency). Un sac de mot est un dictionnaire comportant l'occurence des mots dans le message. L'application de TF (Term Frequency) permet de normaliser leur norme en fonction du nombre total de mots. On applique ensuite le facteur multiplicatif nommé IDF qui relativise la fréquence des mots en fonction de leur présence dans les autres messages du corpus. En effet, si un mot apparaît souvent dans un document mais qu'il apparait par ailleurs couramment dans de nombreux autres documents, ce terme n'est pas discriminant pour le corpus considéré. Le texte est alors catégorisé avec un classifieur traditionnel comme SVM sur les caractéristiques ainsi extraites.

Maintenant, nous tenterons de partir de ces informations pour implémenter notre propre version du `bag-of-words`.

III. Approche proposée

Trois approches ont été adoptées au cours de la recherche. Toutes les trois utilisent une approche dérivant du `bag-of-words` initial. Voici la notation utilisée pour les formules :

N : nombre de discours
D : nombre de mots gardés après le fitrage
V : vecteur discours, de dimension Dx1
n : taille du n-gram
d : indice pour les discours
v : indice pour les mots/n-gram
$f_{v,d}$: fréquence du mot v pour d
$f_{v,t}$: fréquence du mot v au total

La première approche consiste à traduire chaque discours en une représentation `bag-of-words` et de couper les mots les moins communs et les plus communs avec deux paramètres que nous nommerons α et β qui constitue respectivement la borne basse et la borne haute de notre seuillage. Ainsi, l'hypothèse que nous faisons est que les mots les plus populaires ne sont pas intéressants pour le classement, ceux-ci étant principalement des prépositions, des conjonctions, etc. De plus, pour la raison inverse, nous pensons que les mots les moins populaires seront trop rares pour être utile au classement. Les hyperparamètres à trouver ici sont α et β

Deuxièmement, nous tentons d'étendre le concept du `bag-of-words` en utilisant cette fois des N-Gram. C'est-à-dire, utiliser des suites de mots à la place de mots seuls. Nous filtrons cette fois les N-Gram trop rares et trop fréquents et non les mots seuls.

Pour les deux premières parties, les vecteurs V de chaque discours sont construits de la façon suivante :

$$V = [f_{1,d}\ f_{2,d}\ f_{3,d} \cdots f_{D-1,d}\ f_{D,d}]$$

Finalement, nous utilisons une équation semblable aux familles de formules TL-IDF qui pondère en fonction de la rareté du mot. Notre hypothèse ici est qu'un mot plus rare décrit plus précisément l'idée d'un texte qu'un mot plus courant. L'équation 1 montre la pondération utilisée. Nous introduisons un paramètre γ ici pour contrôler la pondération. Nous nommons cet algorithme norm-bow pour le différencier des autres.

$$f_{n,v,d} = \begin{cases} \log \frac{f_{v,d}}{\gamma \cdot f_{v,t}} & \text{si } f_{v,d} \neq 0 \\ 0 & \text{autrement} \end{cases} \quad (1)$$

Le vecteur final prend donc cette forme :

$$V = [f_{n,1,d}\ f_{n,2,d}\ f_{n,3,d} \cdots f_{n,D,d}\ f_{n,D,d}]$$

Ensuite, pour chaque algorithme de `bag-of-words`, nous utilisons un classifieur de type `SVM linéaire` pour l'entraînement en classification. Nous avons utilisé notre propre implémentation des `bag-of-words` que l'on peut retrouver dans le dossier `code` du projet.

IV. Méthodologie expérimentale

La méthodologie expérimentale comporte trois volets : la collecte des données, la construction du `bag of words` et la recherche des hyperparamètres associés aux classifieurs.

Collecte des données La collecte de données a été une étape cruciale pour le bon déroulement du projet. En effet, la qualité d'un algorithme d'apprentissage dépend directement de la diversité et de la quantité des données. Dans le cas présent, les discours politiques ont été prélevés sur le site officiel de l'Assemblée nationale [1].

Les partis politiques que nous avons choisis sont le *Parti Québécois* et le *Parti Libéral du Québec*. Étant donné que ces discours se présentent sous la forme d'échanges verbaux avec interruptions et que chaque discours débute différemment, la collecte des discours a dû se faire manuellement avec un jugement humain avant de les choisir. Pour respecter un minimum de diversité dans les discours et pour recueillir les données en un temps raisonnable, un nombre de 100 discours a été fixé. Une fois le nombre de discours fixé à 100, la répartition des discours a été faite comme suit : pour chaque parti politique, dix députés appartenant à ce parti ont été choisis. Ensuite, pour chaque député, 5 discours ont été sélectionnés à travers le journal des débats pour les mettre dans un fichier texte. À la fin de cet exercice, 100 discours ont été sélectionnés au total (5 discours par personne et 10 personne par parti politique). Pour chaque député, nous avons choisi des discours de taille raisonnable et des sujets variés. Nous avons aussi respecté la parité des sexes pour former un jeu de données plus diversifié

1. http://www.assnat.qc.ca/fr/travaux-parlementaires/journaux-debats/index-jd/41-1.html

Une fois les discours de chaque député rassemblés dans un fichier texte différent, un travail de conversion des données est nécessaire pour fournir des données traitables aux classifieurs `bag-of-words`. Le travail effectué a été d'écrire des fonctions qui, à partir des fichiers contenant tous les discours, éliminent les caractères qui ne sont pas intéressants, enlève toutes les majuscules et sépare le tout en liste de mots. Un fichier `.csv` est généré avec les données traitées qui peuvent être facilement chargées par les algorithmes ensuite.

Construction des bag-of-words Pour construire les `bag-of-words` lors des deux premières approches, nous faisons d'abord un dictionnaire avec les fréquences de chaque mot ou n-gram dans le référentiel global ($f_{v,t}$) de tous les discours. Ensuite, dans chaque discours, nous coupons les mots ou n-grams qui ne respectent pas la condition suivante :

$$f_{v,t} \geq \alpha \quad \text{ou} \quad f_{v,t} \leq \beta$$

Nous calculons ensuite la fréquence du mot ou du n-gram dans le référentiel local du discours ($f_{v,d}$). Nous traduisons ainsi le vecteur de mot en un vecteur V de taille Dx1 pour chaque discours. Pour ces algorithmes, les hyperparamètres à optimiser seront α, β ainsi que la taille d'un n-gram n.

Pour la troisième approche, nous ne filtrons pas les mots ou n-gram, mais utilisons la normalisation par fréquence globale qui donne un poids plus fort aux mots plus rares. Nous n'aurions donc pas besoin de filtrage. Le paramètre γ est nécessaire pour ajuster la force de la régularisation. Pour cet algorithme, les hyperparamètres à optimiser sont la taille des n-grams ainsi que γ.

Optimisation des hyperparamètres Une fois le `bag-of-words` transformé en jeu de donnée

numérique où chaque discours est un vecteur de dimension D, nous pouvons commencer l'optimisation des hyperparamètres pour chaque algorithmes. Nous commençons par faire une division aléatoire du jeu de donnée en partition d'entraînement et en partition de test 50%/50%. Nous utiliserons la partition d'entraînement uniquement pour mesurer la performance des hyperparamètres lors de l'optimisation et pour calculer le score en entraînement à la toute fin. La partition de test sera utilisé seulement à la fin pour calculer le score en généralisation.

Nous faisons une optimisation des hyperparamètres suivants pour les deux premiers algorithmes :

$$\alpha = \{0\ 1\ 2\ ...\ 28\ 29\}$$

$$\beta = \{20\ 21\ 22\ ...\ 68\ 69\}$$

Nous rajoutons cet hyperparamètre pour le deuxième algorithme :

$$n = \{2\ 3\ 4\ 5\}$$

Nous faisons une optimisation sur ces hyperparamètres pour le troisième algorithme :

$$n = \{1\ 2\ 3\ 4\ 5\}$$

$$\gamma = \{0.01\ 0.1\ 1\ 10.0\ 100.0\}$$

Pour chaque couple d'hyperparamètre, afin de mesurer le mieux possible le score en généralisation, nous utilisons une technique de partitionnement de sous-entraînement/validation croisée 5x2 plis, puisque c'est une technique qui est prouvé statistiquement efficace pour faire ressortir l'information de généralisation.

Finalement, le choix des valeurs des paramètres présentés ci-haut se fait grâce à une recherche par grille sur tous les couples possibles (α doit être plus petit que β). Pour comparer les performances des hyperparamètres, nous utilisons le taux de classement comme métrique.

Il est à noter que nous ne ferons pas d'optimisation pour le SVM Linéaire pour cette étude, car nous nous intéressons ici principalement à la représentation des discours avant de passer dans un algorithme de classement.

V. Résultats

Dans le tableau 1, nous retrouvons les meilleurs scores en validation par méthodes. Comme décrit dans la section IV, chaque score en validation représente la moyenne d'un score calculé à partir de la méthode 5x2 plis. À la figure 1, nous pouvons voir un exemple de recherche d'hperparamètre pour α et β pour le `bag-of-words` (BoW) de taille de n-gram $n = 1$.

Tableau 1: *Meilleurs BoW par version (validation)*

Méthode utilisée	taux de classement
Bag-Of-Words	
$\alpha = 18/\beta = 53$	0.98
N-Gram-Bag-Of-Words	
$n = 2/\alpha = 9/\beta = 42$	0.92
$n = 3/\alpha = 2/\beta = 51$	0.89
$n = 4/\alpha = 2/\beta = 51$	0.82
$n = 5/\alpha = 2/\beta = 51$	0.75
Norm-Bag-Of-Words	
norm $n = 1/\gamma = 100.0$	0.75

À partir du tableau 1, nous pouvons choisir les meilleurs méthodes (en bleu) de chaque `bag-of-words` et calculer le vrai score d'entraînement et de généralisation sur la partition d'entraînement et de test respectivement. Ces résultats sont compilés dans le tableau 2.

À première vue, il semble que l'algorithme avec $n = 2$ soit plus fort que l'algorithme. Nous pouvons tout de suite remarquer que notre

Figure 1: *Exemple d'optimisation pour α et β*

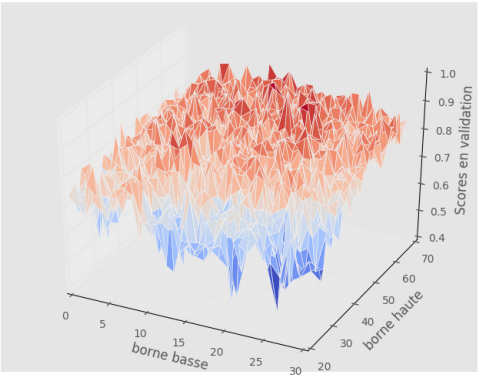

Tableau 2: *Compilation des scores 50/50 des meilleurs BoW*

Méthode utilisée	score en train/test
$\alpha = 18/\beta = 53$	1.0/0.82
$n = 2/\alpha = 9/\beta = 42$	1.0/0.84
norm $n = 1/\gamma = 100.0$	1.0/0.76

normalisation n'arrive pas à rejoindre les performances des méthodes classiques. Nous analyserons les résultats grâce à un test ANOVA dans la section suivante pour voir s'ils sont significatifs et nous discuterons des résultats finaux.

VI. Analyse et discussion

Dans la section V, nous avons montré les taux de classement obtenus en validation ainsi qu'en entraînement et en test. Nous voyons qu'en entraînement, le SVM linéaire n'a aucune misère à apprendre le jeu de données. La première observation intéressante est qu'un modèle qui tente de prendre des groupes de mots semblent performer de meilleure manière que les autres. En effet, notre modèle avec $n = 2$, c'est à dire prenant des groupes de mots de longueur 2 semble obtenir les meilleures performances. Or, il est fort probable que statistiquement parlant, les performances à $n = 1$ et $n = 2$ soient trop proches pour que nous puissions conclure. Dans cette section, nous effectuerons un test statistique pour montrer si la différence entre les algorithmes est significative ou non. Ensuite nous discuterons de l'expérience et des améliorations possibles.

Test ANOVA Pour effectuer le test ANOVA, nous recréons les conditions de validation en entraînant les meilleures versions des algorithmes sur 3 plis. À partir des scores de validations, nous faisons un test ANOVA et arrivons aux résultats compilés dans le tableau 3

Tableau 3: *Compilation des p-values avec AVOVA*

comparaison	p-value
1 vs 2	0.9
2 vs 3	0.7
1 vs 2 vs 3	0.9

Ainsi, nous voyons que, statistiquement parlant, il n'y a pas de différence significative entre les performances des algorithmes. Pour ce faire, nous nous attendions à une valeur au moins < 0.01 de p-value. Nous sommes très loin de la valeur escomptée.

Réflexion sur les résultats Même s'il n'est pas possible de conclure qu'un algorithme est meilleur que l'autre, nous pouvons voir qu'un simple BoW avec modèle basé sur une longueur d'un mot a quand même des performances impressionnante. En effet, nous arrivons à avoir un taux de classement d'environ 80.0% en généralisation, ce qui n'est pas négligeable.

Mots importants les plus utilisés Une analyse intéressante que nous pouvons faire après cela est d'aller voir les mots les plus

fréquents par parti politique dans la tranche choisie par notre optimisation.

Tableau 4: *Mots les plus populaires par parti*

parti	mots	fréquence
PLQ	'compagnies'	37
	'familles'	35
	'engagements'	35
	'choix'	33
	'port'	33
PQ	'immigration'	43
	'professionnels'	43
	'dispositions'	41
	'changements'	38
	'protection'	37

Libre au lecteur de faire ses propres conclusions, mais il est intéressant de voir certaines polarisations dans les mots utilisés par les différents partis. Il ne faut pas oublier que ce travail fut réalisé lors d'un travail de session et que même si nous avons fait un travail de sélection pour les discours, un biais intrinsèque de sélection fait que certains sujets sont plus représentés que d'autres. Dans le dossier `raw_data` le lecteur peut retrouver la provenance de chaque discours ainsi qu'un bref titre décrivant le sujet du texte.

Le lecteur pourra également retrouver un ouvrage très intéressant d'analyse statistique des mots de la politique du Québec écrit par *Dominique Labbé* et *Denis Monière* intitulé *Les mots qui nous gouvernent*.

Amélioration Une amélioration possible consisterait à utiliser une représentation des discours de type `WordsToVec` et de prendre un réseau de neuronnes pour apprendre à prédire l'appartenance du discours. Les chercheurs de chez Google travaillent en effet sur ce type de représentation de données et ont beaucoup de succès dans les compétitions visant à classer des mots difficiles [2].

Références

[1] Georgios Paltoglou and Mike Thelwall. More than bag-of-words : Sentence-based document representation for sentiment analysis. In *RANLP*, 2013.

[2] E. S. Tellez, S. M. Jiménez, M. Graff, D. Moctezuma, R. R. Suárez, and O. S. Siordia. A simple approach to multilingual polarity classification in twitter, December 2016.

2. https ://arxiv.org/pdf/1301.3781v3.pdf