Classification de tweets politiques. Exploration sur la campagne présidentielle de 2017

par Zakarya Després

Encadrant et directeur
de mémoire : Clément Plancq

Mémoire de recherche soumis
dans le cadre du **Master Traitement Automatique des Langues**
Parcours Ingénierie Multilingue
du Département Textes, Informatique, Multilinguisme de l'**Institut National des Langues et Civilisations Orientales**
pour l'année universitaire 2016-2017

inalco
Institut national des langues et civilisations orientales
Lattice

Classification de tweets politiques. Exploration sur la campagne présidentielle de 2017 — Très rapidement après sa création en 2006, Twitter est devenu une plateforme pour la conversation politique, servant à la fois de média de communication pour les personnalités politiques, d'espace de discussion entre militants, et de moyen pour n'importe quel citoyen lambda d'essayer de communiquer avec la personnalité politique de son choix. C'est de ce dernier aspect de la conversation politique sur Twitter que nous avons traité dans le travail ici présenté. Dans un corpus de réponses à des tweets de candidats à l'élection présidentielle de 2017 postés pendant la campagne, nous avons tenté une classification automatique de leur polarité en comparant différents algorithmes. Parmi les choix que nous avons fait pour créer notre modèle, nous avons notamment essayé d'y intégrer des méta-données offertes par Twitter, afin d'inclure le plus d'informations possibles qui nous serviraient nous-même à comprendre le sens d'un tweet. — Zakarya Després, Institut National des Langues et Civilisations Orientales, 2017.

Remerciements — Je tiens à remercier toutes les personnes ayant aidé à la création de ce mémoire. Merci à Clément Plancq, directeur et encadrant de ce mémoire, pour son soutien et sa patience sans égal. Merci à toute l'équipe du Lattice, en particulier à Isabelle Tellier pour son aide précieuse pour comprendre le fonctionnement de Weka et des SVM, et à Thierry Poibeau pour avoir relu ce mémoire. Merci à Julienne Richard, qui a réalisé la mise en page de ce mémoire.

Table des matières

Introduction

Très rapidement après sa création en 2006, le service de *micro-blogging* Twitter a servi de média de communication politique. Dès 2007, l'équipe de relations publiques de Barack Obama a créé le compte @BarackObama pour pouvoir faire la promotion de leur candidat tout au long de la campagne, puis pendant ses huit ans de mandat.

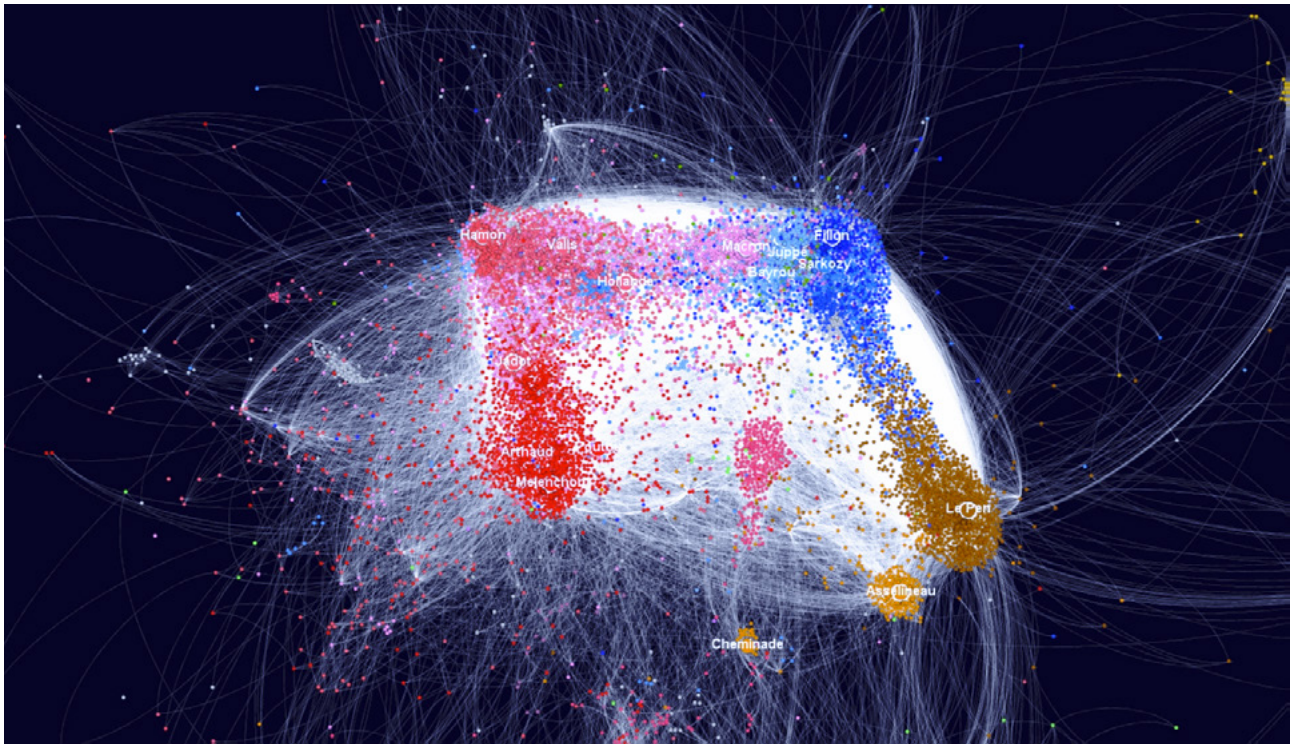
Le réseau social, sous forme de messages de 140 caractères maximum[1] appelés *tweets*, offre une vraie plateforme de communication entre une personnalité publique et sa communauté, que ce soit dans le monde du spectacle ou dans le monde politique. Contrairement aux événements ponctuels que propose la communication traditionnelle des débats télévisés et des conférences de presse, son format ressemble plus à un flux, et permet d'être constamment en contact avec son public. Il donne lieu à une communication du quotidien, où chaque instant mineur de la campagne politique d'un candidat pourra être commenté par ses *followers*, comme on appelle les personnes abonnées à son compte.

1. Depuis novembre 2017, la limitation est désormais de 280 caractères.

D'ailleurs, à l'instar d'autres plateformes sociales comme Facebook, ou plus récemment Discord, Twitter est un espace où des communautés se rassemblent autour d'un intérêt commun pour pouvoir en discuter et organiser des événements en rapport avec celui-ci, parfois en collaborant ou en se heurtant avec d'autres communautés. Dans l'univers de l'internet politique, elles se forment autour de partis politiques, de personnalités particulières au sein des partis quand vient le moment des primaires, ou bien autour de concepts ou d'idéologies plus généraux comme l'écologie ou le syndicalisme.

Ce sont les conversations de ces communautés que le projet Politoscope de l'Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF), mené par David Chavalarias, a tenté d'analyser dans le cadre des élections présidentielles de 2017. Après avoir établi une liste de tous les comptes ayant une affiliation politique, c'est-à-dire ceux des personnalités politique mais aussi ceux des partis ou d'association de partisans, l'intégralité

de leurs tweets a été recueillie sur la période de la campagne, ainsi que les réponses d'utilisateurs lambda leur ayant été adressées. L'ISC a ensuite observé par qui les tweets de chaque compte ont été *retweetés*, c'est-à-dire partagés tels quels, afin d'établir des réseaux de communautés organisés autour des grands partis et des candidats de l'élection présidentielle (Gaumont et al., 2017).



fig, a - Visualisation de la Twittosphère politique, chaque point correspondant à un compte Twitter

Le Lattice, partenaire du Politoscope, souhaitait pouvoir ajouter un niveau d'analyse de ces conversations grâce à leur contenu textuel, plutôt que de n'exploiter que leurs méta-données. Le projet qui est né de cette collaboration et qui a donné lieu à ce mémoire est une tâche de classification de tweets par sentiment dans le domaine de la conversation politique sur Twitter.

Son objectif est d'annoter une série de tweets provenant d'un corpus de réponses à des tweets postés par les candidats à la campagne présidentielle, puis tenter de les classifier grâce à des algorithmes d'apprentissage automatique en fonction de leur polarité, c'est-à-dire en fonction de s'ils expriment des sentiments positifs ou négatifs.

Tweet du candidat François Fillon	Pour enfin améliorer l'accès des jeunes à l'emploi, faisons de l'alternance une voie d'excellence.
Réponse positive	L'alternance est nécessaire. Donc après la gauche en 2012, la droite (républicaine) en 2017 ! :)
Réponse neutre	J'ai un beau CV grâce à mon BTS en alternance
Réponse négative	A propos d'accès à l'emploi @FrancoisFillon, quand comptez-vous exercer une vraie profession ?

tab. a - Exemples de tweets polarisés

Ce type de classifieur peut avoir plusieurs utilités : d'abord, il pourrait permettre d'évaluer la popularité d'un candidat par rapport au ratio de réponses positives et négatives à un tweet, ce qui pourrait d'ailleurs servir à faire des prédictions à propos du résultat des élections. Sur le plan de la recherche, il serait possible d'établir un corpus de tweets politiques positifs et négatifs, qui pourront conduire par exemple à des études textométriques sur le discours des militants.

Si cette tâche a déjà été réalisée avec d'autres types de données ou même avec des tweets, cette fois-ci le but est d'étudier le genre précis du tweet politique, et d'offrir une analyse se basant à la fois sur le contenu textuel du tweet et sur les méta-données qu'il peut proposer.

État de l'art

L'analyse de sentiment, au centre de ce projet, est une tâche classique du traitement automatique des langues : il s'agit de la classification automatique de documents en fonction du sentiment qu'ils expriment. Si elle peut traiter un vaste panel d'émotions comme la joie, la tristesse ou le dégoût, nous nous concentrerons ici sur la seule polarité des tweets de notre corpus.

Parmi les articles fondateurs de cette discipline, on cite volontiers celui de Peter Turney publié en 2002. Les documents à classer sont des avis notés que des internautes ont laissé sur le site spécialisé Epinions. Ils traitent de catégories variées allant de l'automobile aux films, en passant par les banques. Turney en a extrait des expressions grâce à des patrons morpho-syntaxiques, puis a évalué leur "orientation sémantique" grâce à l'algorithme PMI-IR : il se sert de l'information mutuelle pour déterminer la relation entre deux mots, c'est-à-dire de leur corrélation l'un avec l'autre. Dans son cas, il mesure l'information mutuelle entre les expressions et les mots "*excellent*" et "*poor*", qui sont respectivement représentatifs des polarités positives et négatives. Une fois cette étape réalisée, pour classifier un document donné, il suffit alors de calculer la somme des scores des expressions afin d'obtenir l'orientation sémantique globale du test. Cette méthode va donc donner des résultats en valeurs continues, qui vont rendre compte de la polarité d'un document et à quel degré.

Publié en 2002, très peu de temps après Turney, un article de Bo Pang et Lillian Lee a lui aussi été repris de nombreuses fois dans la littérature au sujet de l'analyse de sentiment. Comme pour leur prédécesseur, le corpus est constitué d'avis d'internautes notés entre 1 et 5, cette fois-ci tirés du site de référence sur le cinéma *IMDB*. En séparant les avis positifs des avis négatifs, Pang et Lee ont établi une liste de vocabulaire pour chaque polarité qui semblait être représentative du lexique de chaque type d'avis. Grâce à ces listes, ils ont pu déterminer une baseline en classifiant les avis en fonction des mots du vocabulaire établis qui étaient présents

dans les documents. Ils ont ensuite comparé les résultats obtenus avec différents choix d'attributs dans le modèle d'entraînement et d'algorithmes de classification automatique. Nous allons fortement nous inspirer de cette approche complète et méthodique pour ce projet, pour essayer d'optimiser nos résultats au mieux.

Pour notre tâche, nous travaillerons sur les tweets, un format particulier de texte sur lesquels travailler puisqu'ils sont souvent assez bruités : cela signifie qu'ils ne respectent pas systématiquement les règles du français standard. Pour les utilisateurs lambdas de Twitter, c'est un média informel, où il n'est pas nécessaire de respecter les règles de grammaire et d'orthographe. À l'époque de sa publication, les tweets n'étaient limités qu'à 140 caractères, ce qui est aussi à l'origine d'un peu de bruit, puisque pour la respecter elle peut mener l'auteur d'un tweet à avoir recours à des abréviations inhabituelles, en omettant certaines voyelles par exemple.

Government confirms blast n nuclear plants n japan...don't knw wht s gona happen nw...

fig. b - Exemple de tweet bruité (Ritter, 2011)

Sur un autre niveau, cette limitation ne permet pas de mettre le contenu d'un tweet en contexte. C'est à partir de ces constats qu'en 2011, Ritter a tenté de mettre en place un système de détection d'entités nommées dans les tweets. La nature des tweets lui compliquait la tâche, puisque les tactiques habituelles se montraient bien moins efficaces avec un texte bruité, comme par exemple l'étiquetage morpho-syntaxique du corpus. Afin d'y remédier, il a entre autres réentraîné ses outils d'étiquetage sur des tweets bruités, et a normalisé les différentes variantes orthographiques d'un même mot en les rassemblant en *clusters*, c'est-à-dire en ensembles de mots proches. Malgré cela, il n'obtient pas d'aussi bonnes performances avec son classifieur d'entités nommées qu'avec un texte moins bruité.

En 2010, Alexander Pak et Patrick Paroubek ont tenté de déterminer si ces problèmes se rencontrent aussi pour une tâche d'analyse de sentiment, en proposant un modèle de classification de polarité de tweets. Le corpus, composé de 300000 tweets génériques en français, a été extrait en fonction des smileys qu'ils contenaient : les tweets contenant "*:)*" ou "*:D*" étaient considérés comme étant positifs, et ceux contenant "*:(*" étaient considérés négatifs. Plutôt que de travailler avec une catégorie "neutre", ils ont préféré se servir d'une classe "objectif", avec un corpus correspondant tiré de tweets provenant de divers organes de presse. Ensuite, ils ont comparé les résultats obtenus par différents choix d'attributs avec un classifieur bayésien naïf. Ils ont notamment analysé l'impact d'une représentation du corpus en unigrammes,

bigrammes ou trigrammes, et ont observé que ces derniers donnaient de bien meilleurs résultats. Ils ont aussi mesuré les résultats obtenus avec différentes tailles de corpus d'entraînement : selon leur article, les meilleurs scores s'obtiennent à partir d'un corpus de 150000 tweets, et ne s'améliorent que marginalement avec des corpus de 200000 à 300000 tweets.

D'autres ont déjà défriché le problème de l'analyse de sentiment sur Twitter, comme l'article de Barbosa & Feng (2010) ou les travaux réalisés dans le cadre des conférences DEFT de 2015 et 2017, qui proposaient entre autres une tâche de classification de tweet par polarité. Cependant, ils n'observent jamais le tweet au delà de son aspect textuel, alors qu'il possède de nombreuses méta-données à exploiter. Par exemple, beaucoup de tweets font partie d'une chaîne de réponses faisant écho les unes aux autres, ce qui pourrait apporter un contexte supplémentaire pour leur analyse. En parallèle, d'autres domaines de recherche sont dans une dynamique inverse, en n'utilisant que les méta-données sans prendre en compte le contenu textuel du tweet, comme dans le domaine de la détection de communautés (Barberá, 2010).

Nous souhaitons donc nous positionner entre ces deux écoles, en tentant d'exploiter toutes les données que le format du tweet peut apporter. La méthodologie que nous avons appliqué à la création de notre modèle de classification est la même que celle de Pang et Lee ou de Pak et Paroubek : nous avons tenté d'optimiser notre classifieur en comparant les résultats obtenus par différentes combinaisons d'algorithmes et d'attributs, en tentant d'intégrer à ces derniers des méta-données pour enrichir l'information apportée par chaque tweet.

Description du corpus
— Recrutement des données

En février 2017, au début de ce projet, la campagne présidentielle commençait à battre son plein. Avec pour objectif d'observer les communications des communautés politiques sur Twitter, l'ISC-PIF a recueilli de nombreux tweets écrits en réponse à ceux de différentes personnalités politiques impliquées dans la campagne, des candidats aux députés en passant par divers groupes de soutien.

Nous comptions sur ce vivier de tweets pour construire le corpus, mais à cause d'implications légales liées à la protection des données utilisateurs, ils n'ont pas pu être mis à notre disposition, et il nous a fallu recruter des tweets en rapport à la campagne politique par d'autres moyens.

Un tweet contient de nombreuses données le concernant :
- Le contenu textuel
- La date à laquelle il a été posté
- Le pseudo et le nom de compte de son auteur
- Un numéro d'identifiant
- Les autres tweets qui lui sont adressés en réponse
- Les utilisateurs à qui le tweet répond, le cas échéant
- Le nombre de retweets et de likes : équivalents à des partages, ils mesurent l'engagement des utilisateurs de Twitter envers ce tweet
- Les images qu'il contient
- Sa langue, sa géolocalisation...

Nous avons décidé de ne conserver que les six premiers éléments de cette liste, puisque c'était le minimum nécessaire pour notre analyse. Si nous avons choisi de mettre de côté le nombre de *retweets*, c'est que nous souhaitions nous concentrer sur l'aspect textuel dans un premier temps, puis nous voulions nous détacher des méthodes de data science ne se basant que sur ces informations (Barbera, 2015). Nous portions quelques espoirs sur la présence ou non de *memes*, ces images et gifs drôles et/ou moqueurs, pour nous aider à classifier les réponses. Malheureusement, nous en avons trouvé très peu dans les réponses aux candidats, nous avons donc abandonné cette piste.

La manière la plus simple de récupérer des tweets et toutes ces informations qui lui sont liées est de passer par l'API REST de Twitter. Pour s'en servir, il faut passer par une authentification Oauth via la création d'une application Twitter. Cette dernière possède un jeton d'authentification, qui va nous permettre de faire des appels via les différentes méthodes de l'API.

Pour pouvoir interagir avec elle, nous nous sommes servis de la librairie Tweepy[2], en Python, qui permet de récupérer les tweets et les informations nous intéressant grâce à des fonctions simples à exploiter dans le cadre d'un script Python. Ainsi, nous avons récupéré tous les tweets de chaque candidat à l'élection présidentielle pendant la campagne. Pour chacun de ces tweets, nous avons aussi extrait leur numéro d'identifiant de tweet pour pouvoir y accéder plus tard, ainsi que leur date d'émission.
Tous ces documents ont été conservés au format JSON, plus léger que le XML et plus facile à exploiter dans un script Python.

2. http://www.tweepy.org/


```
{
  "JLMelenchon": [
    {
      "text": "Gagner pour faire comme les autres, ça ne sert à rien. #TF1 #DemainPresident",
      "id": 852950149559193600,
      "date": "2017-04-14",
      "user": "JLMelenchon"
    }
  ]
}
```


fig c. Exemple de tweet au format JSON

Candidat (@pseudo)	Nombre de tweets collectés
N. Arthaud (@n_arthaud)	841
F. Asselineau (@UPR_Asselineau)	1337
J. Cheminade (@JCheminade)	2293
N. Dupont-Aignan (@dupontaignan)	3021
F. Fillon (@FrancoisFillon)	3197
B. Hamon (@benoithamon)	3242
J. Lassalle (@JeanLassalle)	884
M. Le Pen (@MLP_officiel)	2880
E. Macron (@EmmanuelMacron)	2808
J.-L. Mélenchon (@JLMelenchon)	3210
P. Poutou (@PhilippePoutou)	622
TOTAL	**24334**

tab. b - Tableau récapitulatif de la quantité de tweets collectés

À partir de ces tweets, il est possible de récupérer les réponses qui leur sont adressées. Malheureusement l'API de Twitter est assez limitée sur ce point, puisqu'il n'existe pas de méthode permettant d'obtenir les tweets en réponse à un autre tweet. Une solution que nous avions envisagé envisageable de passer outre serait d'utiliser une méthode de leur API de recherche, qui nous donnerait tous les tweets étant adressés à chaque candidat, puis de faire le lien entre les réponses et les tweets de candidat. Mais là aussi, nous nous sommes heurtés à une autre limitation de l'API : seuls les tweets vieux de moins d'une semaine d'une semaine sont accessible avec un compte classique gratuit, alors qu'il aurait fallu pouvoir couvrir toute la période de la campagne présidentielle.

Pour y remédier, nous avons écrit un script Python pour pouvoir effectuer des captures de données d'écran, ou *screen scraping* en anglais.

Lorsqu'on ouvre un tweet dans sa version web, on peut voir le tweet, l'éventuel média ou le lien attaché, et en dessous les différentes réponses, organisées en fils de discussions. Pour économiser des appels serveurs superflus, Twitter a recours à deux méthodes empêchant de capturer la page immédiatement : l'enroulage des fils de discussion et le chargement progressif des réponses au fur et à mesure du défilement. Ces fonctionnalités font appel à des méthodes AJAX, une architecture permettant de changer l'affichage et le contenu d'une page web de façon dynamique, en ne chargeant des données qu'après une interaction de l'utilisateur, notamment via des exécutions de code JavaScript. Ainsi, si l'on veut obtenir plus de données, il faut simuler le comportement d'un utilisateur en ouvrant le tweet dans un navigateur internet et en faisant défiler la page automatiquement avec Selenium.

fig. d - Exemple de tweet comme affiché dans un navigateur

Selenium[3] est un framework en Java permettant de piloter un navigateur internet à partir d'un script, ce qui permet notamment d'imiter le comportement d'un utilisateur humain tel que cliquer, appuyer sur des touches ou bien bouger le curseur de la souris. Il est également possible

3. http://www.seleniumhq.org/

d'exécuter du JavaScript directement dans le navigateur, afin de modifier l'affichage ou le contenu de la page ouverte, ou bien de se déplacer en son sein.

Pour que le script de screen scraping s'exécute dans un temps raisonnable, il ouvre une sélection aléatoire de tweets de chaque candidat récupéré avec Tweepy plus tôt, grâce à l'identifiant du tweet qui a été récupéré. Ensuite, il défile en bas de la page pendant trois secondes afin de récupérer un maximum de réponses tout en conservant un temps d'exécution réaliste. Dans le cas de la page d'un tweet, la fonction JavaScript permettant d'aller tout en bas de la page ne fonctionne pas quel que soit le navigateur. Pour contourner cet écueil, le script envoie l'entrée de la touche Page Suivante (↡) en boucle. D'ailleurs, dans cette même optique, il a fallu sélectionner aléatoirement 20 tweets à traiter par candidat, pour que le script puisse s'exécuter en plusieurs heures plutôt qu'en plusieurs jours. À partir de ces 220 tweets, nous avons pu extraire 6340 réponses.

Candidat (@pseudo)	Nombre de tweets collectés au total	Nombre de tweets exploités	Nombre de réponses collectées
N. Arthaud (@n_arthaud)	841	20	193
F. Asselineau (@UPR_Asselineau)	1337	20	368
J. Cheminade (@JCheminade)	2293	20	44
N. Dupont-Aignan (@dupontaignan)	3021	20	499
F. Fillon (@FrancoisFillon)	3197	20	811
B. Hamon (@benoithamon)	3242	20	1119
J. Lassalle (@JeanLassalle)	884	20	469
M. Le Pen (@MLP_officiel)	2880	20	621
E. Macron (@EmmanuelMacron)	2808	20	781
J.-L. Mélenchon (@JLMelenchon)	3210	20	921
P. Poutou (@PhilippePoutou)	622	20	513
TOTAL	**24334**	**220**	**6340**

tab. c - Tableau récapitulatif des tweets collectés

Malheureusement, par cette méthode de sélection des tweets, les réponses à certains candidats sont moins bien représentés que d'autres, notamment Jacques Cheminade avec ses 44 réponses. Si ce n'est que le reflet de sa communauté peu active sur Twitter, c'est un élément à prendre en compte lorsqu'on se servira de la relation entre les tweets et leur candidat correspondant dans notre modèle de classification.

— Annotation

Une fois les données rassemblées, il a fallu mener une campagne d'annotation pour pouvoir réaliser une classification supervisée et son évaluation. Sur chaque série, deux annotateurs ont étiqueté exactement les mêmes tweets pour pouvoir comparer leur travail et retravailler l'annotation.

Pour choisir notre outil d'annotation, nous avions plusieurs exigences. Tout d'abord, un tweet peut parfois être difficile à interpréter en dehors de son contexte, il fallait donc pouvoir afficher autant de données possibles à leur propos. Ensuite, pour pouvoir réutiliser facilement les données produites, il fallait qu'il puisse prendre le JSON recueilli en entrée, rajouter l'annotation correcte au niveau du noeud du tweet et nous rendre un autre fichier JSON en sortie. Puisque nous ne trouvions aucun logiciel correspondant à nos besoins, nous avons créé notre propre outil d'annotation en ligne de commande, écrit en Python. Avec des raccourcis claviers, il permet d'étiqueter rapidement les tweets sur les différents niveaux de notre grille d'annotation. De plus, encore une fois grâce à Selenium, il ouvre chaque tweet à annoter dans un navigateur, afin d'avoir tout le contexte nécessaire.

fig. e - Si le tweet était adressé à un membre du PCF, le tweet serait négatif. Puisqu'il est adressé à P. Poutou du NPA, on peut le considérer comme étant positif.

Cette annotation s'est faite en trois temps, d'abord sur une grille d'annotation simple. Chaque tweet est d'abord annoté au niveau de sa polarité :

- positif : si le tweet est en accord avec le tweet ou son émetteur
- négatif : si le tweet est en désaccord avec le tweet ou son émetteur
- neutre : si le tweet n'exprime pas d'opinion claire sur le sujet du tweet ou sur son émetteur.

fig. f - De gauche à droite : tweet positif / tweet neutre / tweet négatif

Certaines grilles d'annotation, comme celle de DEFT 2017, contiennent un niveau de polarité "objectif", indiquant que le tweet énonce des faits objectivement, dans un style journalistique. Puisque notre corpus n'est composé que de réponses d'utilisateurs à des candidats, a priori nous n'observerons pas ce genre de tweet, il semblait donc peu judicieux d'y consacrer un niveau de polarité.

Pour tenter d'aider les classifieurs à étiqueter correctement des tweets se servant de blagues ou de second degré pour faire passer un message dont le fond est à l'opposé de sa forme, nous avons également annoté le "LOL" de chaque tweet. Nous avons annoté grâce à ce terme très généraliste tous les tweets drôles, ironiques ou sarcastiques. Nous avons choisi de rassembler tous ces types de tweets sans distinction puisque ce n'est pas l'objectif premier de cette classification, l'objectif de cette annotation étant de tenter classifier un tweet sarcastique paraissant positif comme un tweet négatif. Cette série n'étant qu'une tentative d'annotation pour se familiariser avec le genre des tweets, la grille et les outils, seulement 10 réponses par candidat ont été annotées pour un total de 110 tweets.

Pour évaluer la qualité de notre annotation, nous nous sommes servis du kappa de Cohen pour calculer l'accord inter-annotateur. Il mesure le degré de concordance entre deux annotateurs, en prenant en compte leur annotation et les résultats d'une annotation qui aurait été réalisée au hasard. Compris entre 0 et 1, plus il est proche de 1, plus les annotateurs sont en accord. Il est calculé de la manière suivante, où A est l'accord observé entre les annotateurs, et B est la probabilité qu'il y ait un accord entre les deux annotateurs.

$$\kappa = \frac{A - B}{1 - B}$$

fig g. - Équation du calcul du Kappa de Cohen

Ce premier essai ne s'est pas montré très fructueux, l'accord inter-annotateur étant d'environ 0,40. Cependant, il a permis de se rendre compte de la difficulté d'annoter la polarité du tweet, surtout avec le genre politique, ses enjeux et les émotions qu'il entraîne. Par exemple, certaines réponses n'avaient aucun rapport avec le tweet du candidat, en n'étant qu'une invective ou une référence à un sujet différent de celui abordé.

fig h. - Tweet sans rapport avec le propos du candidat

C'est pourquoi, dans une deuxième série d'annotation de 100 tweets, un niveau de soutien positif, négatif ou neutre a été ajouté. Il permet d'annoter plus facilement les tweets hors sujet, ou des cas plus marginaux comme des partisans d'un candidat en désaccord avec ses propos. Avec cette nouvelle grille, l'annotation s'est déroulée plus facilement, et l'accord inter-annotateur s'est amélioré vers 0,6 malgré le niveau d'annotation supplémentaire.

fig. i - Tweet exprimant son soutien envers Fillon alors qu'il n'a pas de rapport avec le propos du candidat

fig. j - Tweet désobligeant mentionnant les fameuses «affaires» Fillon sans rapport avec le propos du candidat

Une fois satisfaits de la grille, la dernière campagne d'annotation a pu commencer. 993 tweets ont été traités, soit environ 100 tweets par candidat. Après cette campagne, le faible nombre d'annotateurs a permis de corriger chaque conflit entre les annotateurs manuellement, afin d'obtenir une annotation finale la plus correcte possible.

	Positif	Neutre	Négatif
Sentiments	212	483	298
Soutien	182	569	242
LOL	135	Ø	858

tab. d - Répartition des tweets annotés selon leur polarité

Malgré les efforts fournis, certains problèmes limitent la qualité de notre corpus. Tout d'abord, le temps et les moyens à notre disposition n'ont permis d'annoter que 992 tweets au total par deux annotateurs, ce qui est assez faible par rapport aux milliers de tweets étiquetés automatiquement qui sont habituellement utilisés dans d'autres travaux similaires (Pak et Paroubek, 2010). En revanche, cette petite taille de corpus a permis de vérifier chaque étiquette, ce qui offre une grande qualité d'annotation.

Par ailleurs, il arrivait régulièrement que les discussions dérivent au fur et à mesure d'une conversation entre deux utilisateurs, au point où elle n'a plus rien à voir avec le sujet initial. Il a fallu se poser la question de comment traiter ces tweets : valait-il mieux les écarter pour ne garder que des données hautement pertinentes, ou bien fallait-il les traiter indifféremment du reste des tweets ? Cette dernière option a été retenue afin de refléter le mieux possible la réalité de notre corpus, et potentiellement celle de Twitter.

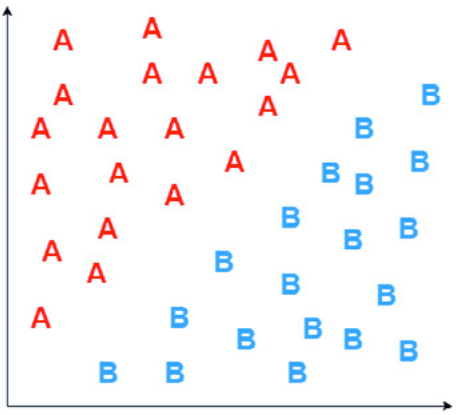
Mais la difficulté ayant eu la plus grande influence sur ce corpus est la complexité d'annoter un tweet politique objectivement. Les opinions de chaque annotateur ont accidentellement une influence sur leur façon d'annoter, et le faible nombre d'annotateurs renforce ce problème en empêchant de mettre en perspective de nombreuses annotations différentes. Dans la continuité de ce phénomène, l'étiquetage au niveau de l'humour implique une grande subjectivité, puisque chacun aura une vision différente de ce qu'il considère drôle ou non. Nous avons tenté de neutraliser ces biais par la comparaison puis la fusion des deux annotations, mais des vestiges des opinions des annotateurs continuent de marquer le corpus.

Classifications
— Classifieurs

Dans le cadre de ce projet, trois types de classifieurs ont été utilisés : les machines à vecteurs de support (SVM), les classifieurs bayésiens naïfs, et les arbres de décision.

Machines à vecteurs de support

Les machines à vecteurs de support, ou SVM pour *support vector machines*, sont des classifieurs binaires conçus par V. Vapnik en 1979.



Leur objectif est de trier des éléments en deux classes par un hyperplan séparateur. Prenons par exemple un plan où se trouvent des éléments A et B, ici représentés respectivement en rouge et en bleu.

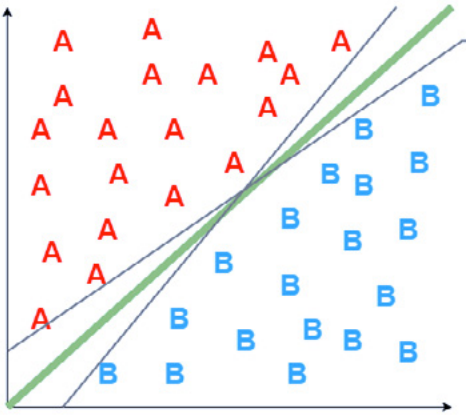
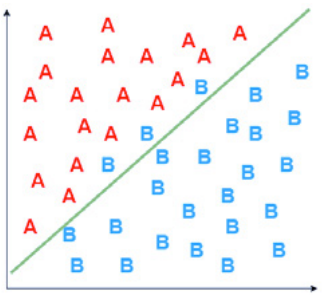


fig. k - En vert, l'hyperplan avec les marges maximales. En gris, des hyperplans possibles mais avec une marge moindre.

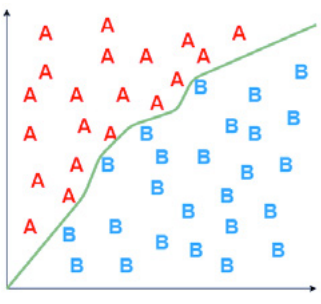
Un SVM veut calculer l'hyperplan avec la plus forte marge, c'est-à-dire qu'il se situe à la plus grande distance possible des éléments à classifier.

Si cet exemple montre un hyperplan droit, on peut paramétrer le SVM pour classifier des éléments qui ne seraient pas séparables de façon linéaire. C'est le rôle du paramètre C, un nombre compris entre 0 et +∞. Plus il est élevé, plus l'hyperplan sera flexible, plus il est faible et plus il sera droit.

Si avoir un hyperplan flexible se rapprochant le plus possible de la véritable répartition des données paraît être la solution idéale pour classifier chaque



C faible



C élevé

élément à la perfection, cela peut créer des phénomènes de surapprentissage. Le modèle donnerait des résultats excellents sur le corpus de travail, mais ne pourrait pas se généraliser à d'autres données similaires. Il faut donc trouver la bonne valeur pour le paramètre C afin d'obtenir un équilibre sain entre la qualité de la classification et sa versatilité.

Puisque *C* pourrait être n'importe quel chiffre positif, il peut être difficile de trouver cette valeur idéale. Pour pallier cela, Schölkopf et al. (2000) ont proposé l'alternative des nu-SVM, qui remplacent le paramètre *C* par le paramètre *nu*. Ce dernier a l'avantage théorique d'être seulement compris entre 0 et 1, ce qui permet de limiter ses valeurs possibles. Cependant, cet avantage ne se traduit pas forcément en de meilleures performances en terme de temps de traitement ou de résultats.

Si en théorie, les SVM ne peuvent réaliser qu'une classification binaire, certaines implémentations comme LIBSVM, utilisée par Weka, permettent un étiquetage multi-classe en effectuant plusieurs classifications dites "seul contre tous". Pour chaque classe, on sépare les éléments qui en font partie des autres, puis le classifieur compile ces différents résultats en une seule classification finale.

Les SVM peuvent se montrer très efficaces par rapport à d'autres méthodes de classification, mais tout dépend des données exploitées. S'ils peuvent rivaliser avec des réseaux neuronaux pour classifier des molécules (Byvatov et al., 2003), ils peuvent aussi devenir moins performants que des classifieurs bayésiens naïfs pour de la classification de texte à partir d'une certaine quantité de données, avec un temps d'entraînement plus long (Colas et Brazdil, 2006).

Classifieurs bayésiens naïfs

Les classifieurs bayésiens naïfs sont des classifieurs probabilistes se basant sur le théorème de Bayes. Ils considèrent qu'il y a une forte indépendance entre les différentes caractéristiques d'une classe : par exemple si une grande part des tweets drôles sont négatifs, le classifieur les considérera comme étant des événements qui n'ont pas d'influence l'un sur l'autre, bien qu'ils semblent être fortement corrélés.

$$P(A \mid B) = \frac{P(B|A)\,P(A)}{P(B)}$$

fig. I - Théorème de Bayes

Un classifieur bayésien calcule la probabilité d'un tweet de faire partie de chaque classe en connaissant ses autres attributs, puis l'étiquette en fonction de la classe ayant la plus haute probabilité.

En dépit de leur apparence simpliste, les classifieurs bayésiens naïfs sont très efficaces, même avec de faibles quantités de données.

Arbres de décision

Les algorithmes se basant sur des arbres de décision effectuent un nombre de choix successifs, que l'on pourrait représenter sous la forme d'un arbre, comme dans le cas suivant d'un classifieur de légumes.

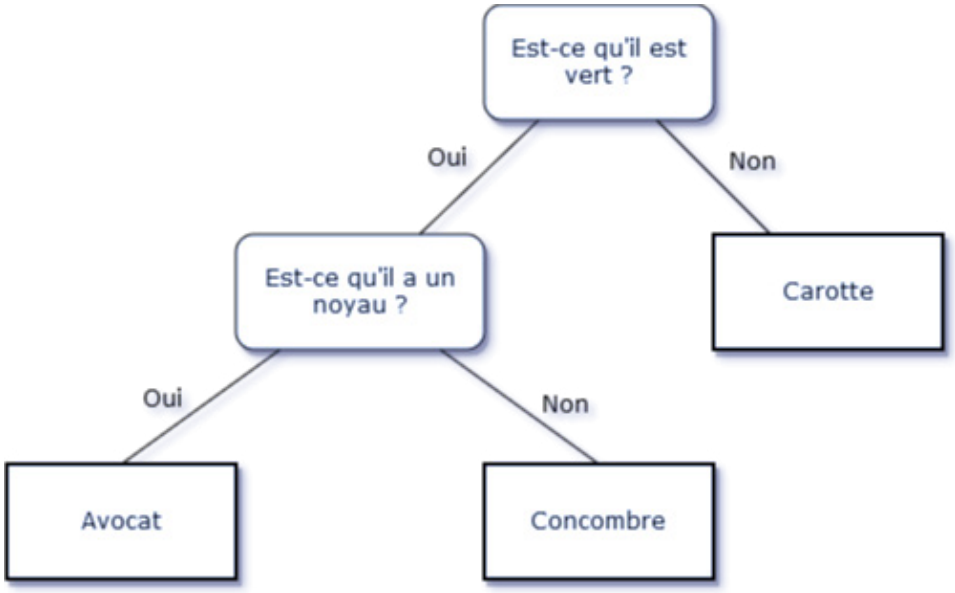


fig. m - Exemple d'arbre de décision classifiant des légumes

S'il existe différents algorithmes de machine learning se reposant sur les arbres de décision, nous nous sommes servis de l'algorithme C4.5 de Ross Quinlan, et plus précisément de J48, son implémentation en Java. Nous avons fait ce choix car c'est un algorithme facile à mettre en place et à paramétrer pour obtenir des résultats probants, c'est pourquoi il est régulièrement utilisé dans ce type d'étude comparative.

À partir du corpus d'apprentissage, C4.5 détermine quel attribut sépare le mieux les données en sous-ensembles significatifs pour chaque classe, en mesurant le gain d'information qu'apporte chaque séparation. Par exemple, si l'on veut classer des fruits et des légumes, un attribut efficace serait de savoir si l'aliment à classer est sucré, alors qu'un attribut inefficace serait sa couleur. Il réitère ensuite la même opération de manière récursive dans chaque sous-ensemble, jusqu'à avoir classé chaque élément.

— Weka

4. https://www.cs.waikato.ac.nz/ml/weka/

Pour réaliser les différentes expériences de classification automatique, nous nous sommes servis du logiciel Weka[4]. Il s'agit d'une suite de logiciels d'apprentissage automatique en Java, créée en 1997 par l'université de Waikato en Nouvelle-Zélande. Avec sa vaste collection d'algorithmes, c'est une solution de référence lorsqu'on veut mettre en place des expériences en apprentissage automatique rapidement et simplement.

Par rapport à d'autres suites équivalentes comme Scikit-Learn en Python, son avantage principal est sa facilité d'utilisation dûe à son interface graphique. En effet, elle permet d'utiliser rapidement différents algorithmes de classification avec des paramètres par défaut. Elle simplifie également l'ajustement de leurs paramètres afin obtenir des classifieurs très performants. Les données que l'on souhaite classifier avec Weka doivent être au format ARFF quel que soit le classifieur utilisé, ce qui permet de facilement comparer les résultats de différents classifieurs avec exactement les mêmes fichiers d'entrée. C'est une norme de format de fichier texte créée spécialement pour la lecture de données dans Weka.

```
@RELATION exemple                       # nom de la relation entre attributs et documents

@ATTRIBUTE attribut1 NUMERIC            # liste des attributs, déclarés avec leur type
@ATTRIBUTE attribut2 NUMERIC            # attribut avec une valeur numérique
@ATTRIBUTE classe {positif,negatif}     # attribut avec une valeur nominale

@DATA                                   # liste des documents à classer
0,0,positif
0,1,negatif
```

fig. n - Exemple de fichier ARFF, avec deux documents à classer

Dans notre cas, les fichiers ARFF auront comme attribut chaque terme apparaissant dans la totalité des tweets du corpus, ainsi que différentes annotations supplémentaires, comme celles de la campagne d'annotation que nous avons menée. Ensuite, chacun de ces tweets sera représenté par une suite de valeurs séparées par des virgules, une par attribut, qui indiquera la présence de chaque terme dans ce tweet ou bien sa catégorie en fonction de l'attribut correspondant.

Expérimentations

Dans ces expériences, les tweets sont classifiés en fonction d'une catégorie donnée, c'est-à-dire en fonction de son sentiment, de son soutien et de son humour, comme défini auparavant. La classification porte donc sur les différentes classes au sein de chaque catégorie, il s'agit donc de positif, négatif, et neutre le cas échéant.

Les multiples tableaux de résultats que vous trouverez au fil de ces pages ont été obtenus avec les configurations d'entraînement proposées par Weka. Nous nous sommes portés sur une validation croisée à 10 plis (Kohavi, 1995), où un échantillon du corpus sert à entraîner le modèle, qui sera testé sur le reste du corpus. Ce procédé est répété dix fois, en changeant l'échantillon d'entraînement à chaque itération.

Lors de chaque évaluation, Weka calcule trois scores pour mesurer l'efficacité de la classification :

- la *précision* : ratio du nombre de tweets correctement étiquetés dans une classe sur le nombre de tweets étiquetés dans cette classe.
- le *rappel* : ratio du nombre de tweets correctement étiquetés dans une classe sur le nombre de tweets appartenant à cette classe.
- la *F-mesure* : moyenne harmonique de la précision et du rappel, elle donne un score de performance global du classifieur.

Si Weka calcule ces trois scores pour chaque classe, il indique aussi la moyenne de chacun d'entre eux pour une seule catégorie. Dans un souci de concision, c'est la moyenne des trois F-mesures pour une même catégorie qui sera indiquée dans tous les tableaux de résultats de ce mémoire, sauf s'il est signalé autrement.

Pour construire notre modèle de classification automatique, nous avons suivi une méthodologie classique. Dans un premier temps, une baseline est établie avec le modèle le plus simple possible, c'est-à-dire qu'on cherche le score minimum qui devra être atteint par nos expériences. Elle donnera une idée de la progression réalisée au fur et à mesure du projet. Ensuite, à partir de ce modèle, des attributs supplémentaires seront ajoutés à ce

modèle basique. En en ajoutant et en vérifiant leur efficacité grâce à la comparaison des résultats obtenus à ceux de la baseline, progressivement un modèle plus efficace sera atteint.
Ce modèle minimal n'a donc aucun attribut de plus que celles du contenu textuel, sans même les annotations des catégories : si on tente de classifier le sentiment d'un tweet, on a aucune information sur son soutien envers le candidat en question. Ce test permet donc, en plus d'avoir une base de travail, d'évaluer les performances des différents classifieurs sur notre corpus lorsqu'ils n'ont que le contenu textuel avec lequel travailler.

	Naive Bayes	SVM	C4.5
Sentiments	0,448	**0,467**	0,429
Soutien	0,518	**0,530**	0,488
LOL	0,799	0,793	**0,799**

tab. e - Résultats servant de «baseline» pour nos expériences. Les meilleures F-mesures sont en gras

On voit ici que si les résultats sont tous dans le même ordre de grandeur, les SVM ont des résultats légèrement meilleurs que les autres algorithmes. Ce n'est pas suffisant pour écarter ces derniers, puisque l'ajout de certains attributs pourrait leur être bénéfique et permettre d'obtenir de meilleurs résultats que les SVM.

Avec ces premiers résultats, on pourrait penser qu'on obtient déjà d'excellents résultats sur la classification humoristique des tweets.
En réalité, les classifieurs n'ont correctement étiqueté que très peu de tweets drôles : le SVM en ont classé 5 sur 136, le Naive Bayes qu'un seul et l'arbre de décision n'en a classé aucun. Le corpus étant principalement composé de tweets pas drôles, le classifieur étiquette la plupart des tweets ainsi et obtient de bons résultats. Pour contrecarrer ce phénomène, il aurait été possible de créer un corpus différent composé à parts égales de tweets drôles et de tweets pas drôles. Cependant, puisque le corpus est déjà de petite taille, cette méthode réduirait encore la quantité de données à disposition. De plus, elle ne reflèterait pas la réalité de la répartition de cette classe dans le corpus. Nous avons donc décidé de conserver cette catégorie en l'état, car malgré son manque de fiabilité pour réaliser un classifieur de tweets humoristiques, elle continue d'être un marqueur mesurable permettant d'évaluer l'impact des différents attributs ajoutées ou retirées pour chaque modèle.

Une fois ce premier essai réalisé, des tests ont été menés pour savoir quel serait la meilleure configuration pour représenter la présence de termes dans les tweets. Trois options ont été considérées :

- représentation *binaire* : indique seulement si le tweet contient le terme correspondant ou non (compris entre 0 et 1)
- représentation par *fréquence* : indique le nombre de fois que le terme correspondant est présent dans le tweet (entre 0 et n)
- représentation par *TF-IDF* : indique l'importance d'un terme dans un tweet par rapport aux autres tweets (entre 0 et n)

En parallèle, nous avons vérifié si les termes devaient être représentés par leur token ou bien leur lemme dans la liste des attributs. Le corpus a été lemmatisé avec MElt, l'étiqueteur morpho-syntaxique de l'équipe ALPAGE de l'INRIA. Ses performances sont celles attendues sur des textes bruités, puisque nous obtenons exactement la même proportion de mots hors-vocabulaire que Ritter dans l'article cité plus tôt (Ritter et al., 2011) : d'un étiqueteur avec une précision de 97% sur du texte de presse (Denis et Sagot, 2012), on obtient une précision de 80% avec des tweets.

SVM	**Tokens**			**Lemmes**		
	Binaire	Fréquence	TF-IDF	Binaire	Fréquence	TF-IDF
Sentiment	0,475	0,485	0,491	0,487	0,499	**0,514**
Soutien	0,523	0,536	0,543	0,530	**0,553**	0,551
LOL	0,807	0,808	0,803	0,791	0,788	**0,807**

tab. f - Comparaison des résultats obtenus avec un SVM et différentes représentations de termes avec un SVM

Naives Bayes	**Tokens**			**Lemmes**		
	Binaire	Fréquence	TF-IDF	Binaire	Fréquence	TF-IDF
Sentiment	0,471	0,482	0,426	0,491	**0,489**	0,416
Soutien	0,507	0,503	0,429	0,511	**0,529**	0,390
LOL	0,792	0,793	0,588	0,806	**0,794**	0,545

tab. g - Comparaison des résultats obtenus avec un classifieur bayésien et différentes représentations de termes avec un classifieur bayésien

C4.5	Tokens			Lemmes		
	Binaire	Fréquence	TF-IDF	Binaire	Fréquence	TF-IDF
Sentiment	0,424	0,421	0,435	0,442	0,433	**0,455**
Soutien	0,469	0,477	0,478	0,476	0,479	**0,483**
LOL	0,800	0,800	0,802	0,801	0,801	**0,807**

tab. h - Comparaison des résultats obtenus avec un arbre de décision et différentes représentations de termes avec un arbre de décision

Là encore, les différences entre chaque choix d'attributs sont minces, mais la combinaison d'une représentation avec des lemmes et les TFIDF semblent être en général l'option offrant les meilleures performances, c'est donc celle qui a été conservée.

Le dernier point sur la représentation du corpus dans le modèle à déterminer était celle des n-grams. Au lieu d'entraîner le corpus au niveau du terme seul, on peut décider de travailler avec une partie de son contexte, en y associant le mot suivant voir plus. Pak et Paroubek (2010) ont montré que pour leur tâche similaire, les bigrammes de deux mots obtenaient de meilleures performances que les trigrammes de trois mots ou les unigrammes ne représentant que le terme seul. Cependant, leur corpus était d'une taille bien supérieure au nôtre, il était donc nécessaire de vérifier si ces observations s'appliquent à notre corpus.

	Naive Bayes			SVM			C4.5		
	Unigramme	Bigramme	Trigramme	Unigramme	Bigramme	Trigramme	Unigramme	Bigramme	Trigramme
Sentiment	**0,416**	0,354	0,354	**0,514**	0,359	0,324	**0,455**	0,344	0,318
Soutien	0,390	0,348	**0,422**	**0,551**	0,441	0,419	**0,483**	0,431	0,417
LOL	0,545	0,743	**0,803**	**0,807**	0,804	0,804	**0,807**	0,801	0,801

tab. i - Comparaison des résultats obtenus avec différentes tailles de n-grammes

Au vu de ces résultats, il semblerait que l'utilisation des unigrammes reste la meilleure méthode pour classer ce corpus. À cause de la petite taille du corpus, regrouper les termes en bigrammes ou en trigrammes a tendance à créer des hapax, c'est-à-dire des occurrences uniques de ces n-grammes. Le classifieur ne réussit alors pas à retrouver des éléments communs entre les différents tweets, ce qui l'empêche de correctement les identifier et les classifier.

Une fois que nous avons établi la façon de représenter le contenu textuel pour les classifieurs, il est possible d'ajouter les informations supplémentaires de notre annotation. Par exemple, pour classifier les tweets en fonction de leur sentiment, on ajoute au tweet son annotation de soutien et son annotation d'humour. D'abord, nous n'avons ajouté qu'une seule des deux annotations supplémentaires pour pouvoir mesurer leur impact individuel, avant d'observer leur efficacité conjointe.

	SVM			C4.5		
	Sentiment	**Soutien**	**LOL**	**Sentiment**	**Soutien**	**LOL**
Sentiment		**0,657**	0,526		**0,748**	0,435
Soutien	**0,678**		0,541	**0,732**		0,472
LOL	0,804	**0,807**		**0,807**	**0,807**	

tab. j - Comparaison des résultats obtenus en ajoutant différentes combinaisons de niveaux d'annotation au modèle.

Ces ajouts ont eu un effet très positif sur la classification, notamment pour le classifieur C4.5 dont les performances dépassent largement les SVM lors de la classification de sentiment avec l'aide des informations de soutien et inversement. En revanche, l'annotation sur l'humour seule semble ne pas avoir un fort impact sur la classification, ce qui se confirme lorsqu'on combine toutes les annotations dans le modèle : l'amélioration des scores est faible par rapport à l'ajout des informations de soutien ou de sentiment.

	Naive Bayes	SVM	C4.5
Sentiments	0,453	0,671	**0,745**
Soutien	0,406	0,678	**0,737**
LOL	0,544	0,805	0,805

tab. k - Comparaison des résultats obtenus avec la combinaison de tous les niveaux d'annotation.

Le fait de travailler sur un corpus provenant de Twitter nous offre des informations supplémentaires. En effet, les tweets de notre corpus ne sont pas indépendants : ils font tous partie d'un fil de discussion qui part d'un tweet d'un candidat. De plus, ils peuvent être en réponse à un autre utilisateur de Twitter que le candidat. Nous sommes donc partis de l'hypothèse suivante : le contenu textuel d'un tweet peut être interprété et classifié différemment en fonction du candidat à qui il fait réponse, ainsi en fonction du tweet auquel il répond. Pour tenter d'y répondre, nous avons extrait trois attributs de chaque tweet : le candidat à l'origine du fil de discussion dont il fait partie, l'identifiant du tweet du candidat auquel il répond, et s'il s'agit d'une réponse à une autre réponse.

SVM	Candidat	Identifiant	Réponse à réponse
Sentiments	0,668	0,670	**0,683**
Soutien	0,684	0,682	**0,693**
LOL	0,804	**0,805**	**0,805**

tab. l - Résultats obtenus grâce à un SVM avec l'ajout de différentes méta-données au modèle

C4.5	Candidat	Identifiant	Réponse à réponse
Sentiments	0,738	0,733	**0,753**
Soutien	0,727	0,720	**0,739**
LOL	0,804	**0,807**	0,806

tab. m - Résultats obtenus grâce à un arbre de décision avec l'ajout de différentes méta-données au modèle

Différencier les tweets étant directement en réponse au tweet du candidat et celles de fils de discussion entre utilisateurs semble avoir été le paramètre le plus efficace pour améliorer notre modèle. Nous avons ensuite comparé ces résultats avec ceux qu'on obtiendrait en combinant ces attributs.

SVM	Candidat & identifiant	Candidat & réponse à réponse	Identifiant & réponse à réponse
Sentiments	0,667	0,672	**0,683**
Soutien	**0,696**	**0,696**	0,692
LOL	0,805	0,805	**0,806**

tab. n - Résultats obtenus grâce à un SVM avec l'ajout de différentes combinaisons de méta-données au modèle

C4.5	Candidat & identifiant	Candidat & réponse à réponse	Identifiant & réponse à réponse
Sentiments	0,739	0,735	**0,745**
Soutien	0,714	**0,728**	0,725
LOL	**0,807**	0,804	0,806

tab. o - Résultats obtenus grâce à un arbre de décision avec l'ajout de différentes combinaisons de méta-données au modèle

	Naive Bayes	SVM	C4.5
Sentiments	0,459	0,674	**0,733**
Soutien	0,409	0,697	**0,721**
LOL	0,546	0,804	**0,806**

tab. p - Comparaison des résultats obtenus avec l'ajout de la combinaison de toutes les méta-données au modèle

Là aussi, la séparation des tweets en réponse à une autre réponse et ceux directement adressés aux candidats semble donner de meilleurs résultats. Les paramètres d'identification du candidat et du tweet d'origine semblent être moins impactants, au point de rendre la classification moins efficace si on les associe tous les deux à l'attribut "réponse à réponse".
Cela pourrait dire que tous les tweets d'une même polarité ont de fortes similarités, quelle que soit le candidat ou le sujet auquel ils répondent.

Pour finir, inspirés par les approches hybrides de classification automatique et par lexique des campagnes DEFT 2015 et 2017 (Nzali et al., 2015), nous avons tenté d'incorporer le lexique de polarité Polarimots dans les attributs de notre modèle. Il a été créé par Núria Gala et Caroline Brun en 2012, et est composé de 7483 mots triés en trois catégories de polarité : positif, négatif et neutre. Il a été constitué à partir d'un corpus d'adjectifs dont la polarité a été annotée, puis propagée aux autres mots de leur famille morphologique respective. Par exemple, si *allègre* est annoté comme étant un adjectif positif, sa polarité est propagée aux mots *allègrement* et *allègresse* qui seront positifs également.

Pour représenter cette information dans notre modèle, nous avons utilisé la méthode utilisée par Turney évoquée dans l'état de l'art : un score de polarité est incrémenté pour chaque mot positif contenu dans le tweet, et décrémenté pour chaque mot négatif présent. Puisqu'ils n'ont pas d'influence avec cette méthode, nous avons décidé d'ignorer les mots catégorisés comme étant neutres dans Polarimots. On obtient ainsi un score global pour le tweet qui est utilisé comme attribut dans notre modèle. Dans les résultats suivants, nous nous sommes servis de la configuration ayant obtenu les meilleurs résultats jusqu'ici en moyenne, avec les informations d'identifiant du tweet du candidat et de "réponse à réponse".

	Naive Bayes	SVM	C4.5
Sentiments	0,460	0,678	**0,745**
Soutien	0,408	0,692	**0,722**
LOL	0,550	**0,804**	**0,804**

tab. q - Résultats de l'ajout d'un lexique de polarité au modèle

La configuration de l'algorithme d'arbres de décisions C4.5 avec l'attribut "réponse à réponse" combiné avec l'identifiant du tweet du candidat, sans ajout d'un lexique de polarité, semble donc être la plus performante. En éliminant les scores du LOL qui ne semblent pas être pertinents, on obtient une F-mesure moyenne de 0,735.

Nous allons donc pouvoir comparer nos résultats avec ceux d'autres travaux similaires. Pak et Paroubek ont évaluer leur classifieur sur plusieurs tailles de corpus, c'est pourquoi les résultats à 1000 tweets et à 300000 tweets sont indiquées : les premiers serviront à comparer les résultats à corpus égal, et les seconds pour voir les meilleures performances qu'ils aient atteints. Il faut bien noter que leurs mesures sont différentes des nôtres, puisqu'il se sont servis de la mesure F0,5 alors que nous nous sommes servis de la mesure F1. Nous allons aussi comparer notre classifieur à celui du LIA de la conférence DEFT 2017 (Rouvier et Bousquet, 2017), qui a recours à des réseaux de neurones, et à celui du LIF lors de la conférence de 2015 (Rouvier et Favre, 2015), se basant sur un ensemble de systèmes. Ils étaient tous deux arrivés premiers sur une tâche de classification de tweet par polarité.

Classifieur	Algorithme	Score
Després	C4.5	0,735
LIA DEFT 2017	Réseaux de neurones convolutifs	0,612
LIF DEFT 2015	Plusieurs algorithmes	0,736 (macro-précision)
Pak & Paroubek (1000 tweets)	Naive Bayes	0,41 (F0,5)
Pak & Paroubek (300000 tweets)	Naive Bayes	0,62 (F0,5)

tab. r - Comparatif de nos résultats avec ceux de l'état de l'art

Conclusion

Nous avons pu atteindre des résultats convaincants pour notre classification, notamment grâce aux arbres de décision et aux SVM. Il semblerait que fondamentalement, le choix de l'algorithme ait une plus forte influence sur les résultats que le choix de la représentation des attributs, qui n'ont eu qu'un impact mineur sur les performances des classifieurs.

Cependant, même si les résultats semblent comparables à ceux d'autres travaux plus avancés, la classification souffre de deux limitations pouvant biaiser les résultats. Premièrement, elle se base sur un corpus de faible taille, ce qui risque d'avoir créé des phénomènes de surapprentissage, bien qu'on ait tenté de mitiger cet effet grâce à une validation croisée. Enfin, ces résultats dépendent d'une forte supervision, ce qui peut empêcher le modèle de s'appliquer aussi bien à des données non-annotées et donc de pouvoir se généraliser. Il faudrait donc vérifier notre modèle avec d'autres données.

Pour aller plus loin avec cette tâche, une autre campagne d'annotation devrait être menée, ce qui nous permettrait dans un premier temps de régler ce problème de taille de corpus. Pour commencer à voir à quel point notre modèle est généralisable, on pourrait annoter des tweets d'autres personnalités politiques, afin de voir si notre modèle s'applique aussi à d'autres tweets politiques mais dans un paradigme différent : on pourrait par exemple observer les conversations en dehors du feu de la campagne, ou bien celles en réponse à des représentants du gouvernement plutôt qu'à des candidats. Ensuite, choisir un autre genre de tweets nous permettrait de voir à quel point notre modèle est généralisable : on pourrait par exemple l'essayer avec des tweets en réponse à de grandes marques. Avec ce type de comptes, on peut observer des comportements similaires qu'avec des personnalités politiques, où les utilisateurs soutiennent ou critiquent la marque en réponse à leurs tweets, et pas forcément en rapport avec leur contenu.

Une fois que notre modèle de classification sera bien réglé, que faire de ses résultats ? Plusieurs applications sont possibles : toujours dans la lignée du projet Politoscope, ces opinions pourraient être combinées aux autres informations de communautés pouvoir observer la popularité d'un candidat au fur et à mesure de la campagne, au sein de son propre parti comme chez d'autres. On pourrait également songer à s'en servir pour créer des corpus qui seraient utiles dans des études textométriques, pour observer le vocabulaire positif et négatif d'un instant dans la communication politique.

Bibliographie

Barberá, P., Jost, J. T., Nagler, J., Tucker, J. A., & Bonneau, R. (2015). Tweeting from left to right: Is online political communication more than an echo chamber?. *Psychological science*, 26(10), 1531-1542.

Barbosa, L., & Feng, J. (2010). Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (pp. 36-44). Association for Computational Linguistics

Byvatov, E., Fechner, U., Sadowski, J., & Schneider, G. (2003). Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. *Journal of chemical information and computer sciences*, 43(6), 1882-1889.

Colas, F., & Brazdil, P. (2006). Comparison of SVM and some older classification algorithms in text classification tasks. *Artificial Intelligence in Theory and Practice*, 169-178.

Denis, P., & Sagot, B. (2012). Coupling an annotated corpus and a lexicon for state-of-the-art POS tagging. *Language resources and evaluation*, 46(4), 721-736.

Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for «Data Mining: Practical Machine Learning Tools and Techniques», Morgan Kaufmann, Fourth Edition, 2016.

Gala, N., & Brun, C. (2012). Propagation de polarités dans des familles de mots: impact de la morphologie dans la construction d'un lexique pour l'analyse de sentiments (Spreading Polarities among Word Families: Impact of Morphology on Building a Lexicon for Sentiment Analysis) [in French]. In *Proceedings of the Joint Conference JEP-TALN-RECITAL* 2012, volume 2: *TALN* (pp. 495-502).

Gaumont, N., Panahi, M., & Chavalarias, D. (2017). Methods for the reconstruction of the socio-semantic dynamics of political activist Twitter networks.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai* (Vol. 14, No. 2, pp. 1137-1145)

Nzali, M. D. T., Abdaoui, A., Azé, J., Bringay, S., Lavergne, C., Mollevi, C., & Poncelet, P. (2017). FrenchSentiClass: un Système Automatisé pour la Classification de Sentiments en Français. In *Actes de l'atelier DEFT de la conférence TALN 2017*.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 79-86). Association for Computational Linguistics.

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *LREc* (Vol. 10, No. 2010)

Ritter, A., Clark, S., & Etzioni, O. (2011). Named entity recognition in tweets: an experimental study. In Proceedings of the Conference on Empirical Methods in *Natural Language Processing* (pp. 1524-1534). Association for Computational Linguistics.

Rouvier, M., Favre, B., & Rajendran, B. A. (2015). TALEP@ DEFT'15: Le plus cooool des systèmes d'analyse de sentiment. In *Actes de la 11e Défi Fouille de Texte* (pp. 97-103). Association pour le Traitement Automatique des Langues.

Rouvier, M., & Bousquet, P. M (2017). LIA@ DEFT'2017: Multi-view Ensemble of Convolutional Neural Network. In *24e Conférence sur le Traitement Automatique des Langues Naturelles* (TALN)(p. 13).

Turney, P. D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 417-424). Association for Computational Linguistics.

Vapnik, V. (1979). *Estimation of dependences based on empirical data*. Springer Verlag.

Liste des figures

Liste des tableaux

— suite

Classification de tweets politiques. Exploration sur la campagne présidentielle de 2017 — Zakarya Després, Institut National des Langues et Civilisations Orientales, 2017.