Andréa De Sousa
Chloé Lecointe

Apprentissage automatique : classification

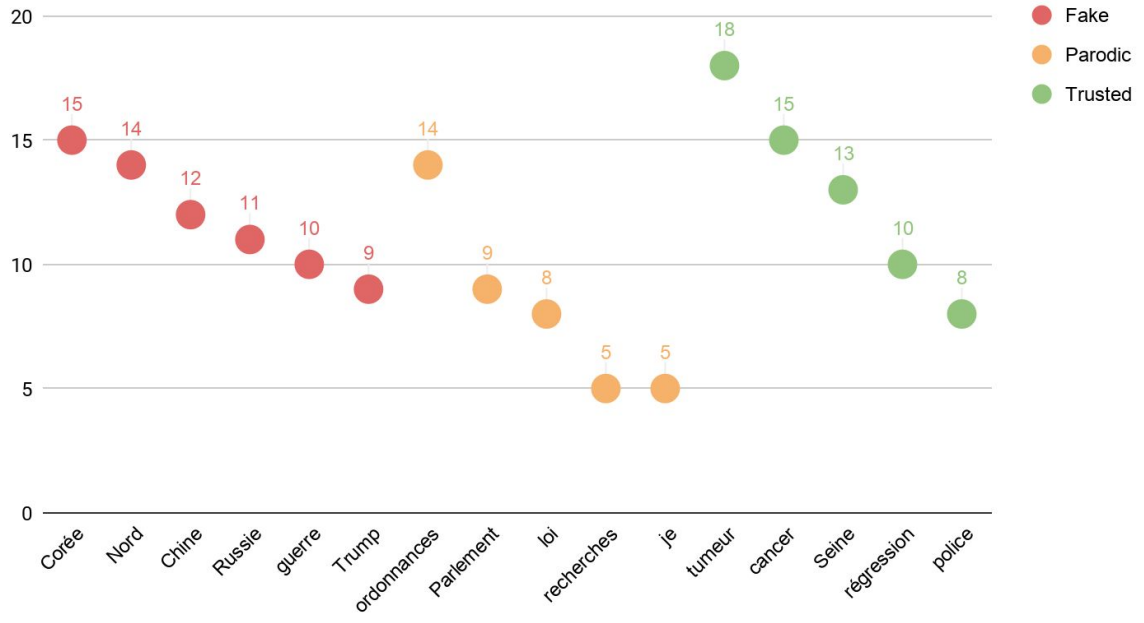
I. Introduction

Afin de déterminer automatiquement si un article de presse, issu de différentes sources disponibles en ligne, est porteur d'informations erronées ("*fake news*"), d'informations intentionnellement erronées ("*parodic news*") ou bien même d'informations véridiques ("*trusted news*"), il est essentiel que le classifieur possède des caractéristiques (*features*) des échantillons pertinents. C'est pourquoi, dans un premier temps, nous allons essayer d'améliorer la prise de décision du modèle en affinant le choix des *features* à l'aide d'outils textométriques comme *Lexico3*. Une fois les *features* sélectionnées, nous allons tenter de déterminer dans un second temps avec *Weka* l'algorithme d'entraînement le plus performant selon nos données d'entrée. Nous établirons enfin dans un dernier temps une évaluation du modèle choisi et présenterons des pistes possibles d'amélioration.

II. Sélection des features

A partir du fichier XML du corpus d'entraînement donné en argument, le script `buildcorpus_lexico.py` introduit les jalons textuels `<category>` (classe de l'article) et `<article>` (identifiant numérique de l'article) qui permettent à *Lexico3* de partitionner les données textuelles afin de pouvoir procéder aux analyses des différentes classes. Une fois les données ré-encodées en ANSI, nous pouvons établir, à l'aide du calcul des spécificités positives par parties, le graphique suivant :

Spécificités des classes fake, parodic et trusted



En observant les spécificités des différentes classes, on peut d'ores et déjà s'apercevoir que la classe "*parodic*" sera la classe la plus difficile pour le modèle à déterminer, puisque ses termes sont globalement moins spécifiques.

Certaines de ces features vont ainsi faire l'objet de fonctions dans `getfeatures.py`.

III. Les algorithmes d'entraînement

Decision Trees - J48

La première expérience est réalisée avec l'algorithme des arbres de décision. Les *features* retenues sont `russia_count`, qui compte le nombre d'occurrences du *token* "Russie", `punct_count`, pour les occurrences des ponctuations de citation, `i_count`, pour les occurrences du pronom personnel "je", et enfin, `adv_count` pour le nombre d'occurrences des adverbes.

Les résultats sont plutôt satisfaisants : sur 49 instances, 42 ont bien été classées, soit un total de 86% de catégories correctement attribuées. La matrice de confusion est la suivante :

	Fake	**Trusted**	**Parodic**
Fake	15	1	0
Trusted	2	13	2
Parodic	1	1	14

On remarque que les erreurs sont relativement moindres : la catégorie "*fake*" n'a jamais été classée comme étant du "*parodic*", et la catégorie "*parodic*", jugée comme la plus difficile à classer lors de la sélection des *features*, a finalement posé moins de problèmes que la catégorie "*trusted*", qui comporte un total de 4 articles classés incorrectement.

Comme le montre le tableau d'évaluation ci-dessous, la classe la plus justement prédite est la catégorie "*fake*", avec une F-mesure de 0,82. Vient juste derrière la catégorie "*parodic*", avec une F-mesure de 0.81, puis, en dernier lieu, la catégorie "*trusted*" avec une F-mesure de 0,73.

	Fake	**Trusted**	**Parodic**
Précision	0.938	0.765	0.875
Rappel	0.882	0.813	0.875
F-mesure	0.824	0.725	0.814

L'arbre de décision sur lequel s'est fondé l'algorithme est le suivant :

```
russia_count <= 1
|   punct_count <= 4: fake (9.0/2.0)
|   punct_count > 4
|   |   punct_count <= 16
|   |   |   i_count <= 0
|   |   |   |   punct_count <= 8
|   |   |   |   |   punct_count <= 6: trusted (4.0/1.0)
|   |   |   |   |   punct_count > 6: parodic (4.0)
|   |   |   |   punct_count > 8
|   |   |   |   |   adv_count <= 37: trusted (3.0)
|   |   |   |   |   adv_count > 37: fake (3.0/1.0)
|   |   |   i_count > 0: parodic (12.0/2.0)
|   |   punct_count > 16: trusted (8.0/1.0)
russia_count > 1: fake (6.0)
```

Bayes - NaiveBayesMultinomial

Une seconde expérience a été réalisée avec un classifieur bayésien naïf. Cette fois-ci, une première ébauche des résultats a montré que pour fonctionner, l'algorithme nécessitait plus de *features* qu'avec un arbre de décision. Celles qui ont été choisies sont les mêmes qu'à l'étape précédente, avec l'ajout de `china_count`, qui compte le nombre d'occurrences du *token* “Chine”, `usa_count`, qui compte le nombre de formes “US” et “USA”, `vous_count`, pour les occurrences du pronom personnel “vous”, et `united_states_count`, qui compte le nombre d'occurrences du *token* “Etats-Unis”.

Comme annoncé, les résultats sont légèrement moins bons mais restent honorables : sur 49 instances, 30 ont bien été classées, soit un total de 61% de catégories correctement attribuées. La matrice de confusion est la suivante :

	Fake	**Trusted**	**Parodic**
Fake	11	3	2
Trusted	1	12	4
Parodic	1	8	7

Comme nous l'avions prédit, la confusion s'est faite majoritairement entre “*trusted*” et “*parodic*”, avec 4 articles véridiques classés comme parodiques, et 8 articles parodiques classés comme véridiques. La catégorie “*fake*” est en général celle qui se comporte le mieux.

Le tableau ci-dessous montre que la classe la mieux prédite est la catégorie “*fake*”, avec une F-mesure de 0,76. Vient ensuite la catégorie “*trusted*”, avec une F-mesure de 0.60, puis enfin, la catégorie “*parodic*” avec une F-mesure de 0,48.

	Fake	**Trusted**	**Parodic**
Précision	0.846	0.522	0.538
Rappel	0.688	0.706	0.438
F-mesure	0.759	0.600	0.483

Pour créer son modèle, l'algorithme a calculé les probabilités suivantes :

	Fake	**Trusted**	**Parodic**
`adv_count`	0.66	0.64	0.62
`china_count`	0.01	0	0
`i_count`	0.01	0.03	0.08
`punct_count`	0.22	0.31	0.23
`russia_count`	0.04	0	0
`united_states_count`	0.03	0	0
`usa_count`	0.02	0	0
`you_count`	0.01	0.01	0.06

IV. Conclusion

Pour finir, le modèle d’apprentissage le plus concluant à l’issu de nos différentes expériences utilise l’algorithme des arbres de décision (*J48*). Toutefois, les résultats peuvent être améliorés. On pourrait rajouter d’autres *features*, ainsi qu’un corpus d’apprentissage plus conséquent afin d’améliorer la prise de décision du modèle. D’autres algorithmes peuvent également être testés sous *Weka*, comme l’algorithme *C4.5* ou *Logistic Regression*.