Apprentissage automatique

Mingqiang Wang Jielei Li Arthur Provenier
yann.losay@gmail.com, happylijielei@163.com, arthur.provenier@gmail.com

1 Bibliothèques requises

Pour lancer les scripts, plusieurs bibliothèques ont été utilisé, cette partie se charge de les énumérer avec leur numéro de version.

- tensorflow, version 1.7.0
- numpy, version 1.14.2
- scikit-learn, version 0.19.1
- matplotlib, version 2.2.2

Si Tensorflow n'est pas installé depuis les sources, des warnings peuvent apparaître, néanmoins ils ne gênent en rien la bonne exécution du script. La compilation des sources permet cependant l'amélioration du temps de calcul du modèle, assez risible sur notre modèle, le temps d'exécution des scripts étant inférieur à 2min.

2 Introduction

« Fake ou pas fake ? », ce qui pourrait être le départ d'un nouveau jeu entre amis, est en fait une problématique très importante suite à l'explosion du web. En effet, de plus en plus de sources journalistes, ou se posant comme telles, voit le jour. Parmi ce flot d'informations il est difficile de distinguer ce qui est vrai ou pas. Le Collins Dictionary [1] définit une fake news comme étant « false, often sensational, information disseminated under the guise of news reporting ». Ainsi, il s'agit d'une news se basant sur des sources non avérées et se voulant plutôt un effet de buzz afin d'attirer le plus de vues possibles ou bien voulant dés-informer. Il est important de ne pas confondre avec les articles se voulant parodique, c'est à dire qui ont pour vocation de tourner en dérision certains sujets et de ne pas se poser en tant que référence.

La distinction entre une news et une fake news semble être difficile même pour Donald Trump, président des États-Unis d'Amérique, qui accuse même CNN qui est un média réputé d'être à l'origine de fake news, « You are fake news » [2]. Il devient ainsi intéressant de pouvoir distinguer parmi plusieurs articles ceux qui pourraient s'apparenter à des fakes news ou non.

Ce projet se donne l'objectif de vouloir prédire automatiquement le type de news à partir d'un corpus préalablement construit en proposant une classification en *fake*, *trusted* ou bien *parodic*.

1. https://www.collinsdictionary.com/dictionary/english/fake-news
2. https://www.usatoday.com/story/news/politics/onpolitics/2017/01/11/trump-cnn-press-conference/96447880/

3 Corpus

Le corpus est constitué d'articles collectés sur différentes sites internet tel que le Monde [3], Réseau international [4], Legorafi [5]. Au total, 148 articles ont été récupéré et seront exploités pour former un corpus d'apprentissage et de test.

Catégorie	Corpus Train	Corpus Test	Total
Fake	42	17	59
Trusted	37	16	53
Parodic	20	16	36
Total	99	49	148

TABLE 1 – Répartition du corpus par catégorie

Tous les articles sont regroupés dans un fichier xml avec deux versions, une version brute obtenue en copier-collant l'article sur le site et une version annotée à l'aide de TreeTagger [6].

4 Méthode employée

4.1 Librairie et algorithme

Afin d'établir la prédiction sur la catégorie de l'article, nous avons choisi d'opter pour la librairie d'apprentissage automatique TensorFlow [7] développé par Google et utilisable via python en l'installant à l'aide de pip.

L'algorithme employé est le *gradient descent* qui nous permet de minimiser la fonction d'erreur en diminuant progressivement son taux d'erreur. Néanmoins, il faut veiller à ce que le pas d'apprentissage ne soit pas trop élevé où le modèle risque de dépasser le minimum et remonter dans la courbe ; à l'inverse, un pas trop faible risque de prendre énormément de temps avant de trouver le minimum. Le bias et le *weight* ne sont pas non plus à négliger. Par la suite, nous ferions varier chacun des paramètres pour tenter d'améliorer notre modèle.

4.2 Features

Les features utilisées par le modèle sont les suivantes : un tf-idf a été réalisé sur l'ensemble du corpus, et les *n* termes avec un score le plus haut seront gardés comme feature, de plus des features personnalisées ont également été ajouté, comme la présence des mots « selon », « nous », « on » et la présence de certains signes de ponctuations comme : ? :,«».

3. https://www.lemonde.fr
4. https://reseauinternational.net
5. legorafi.fr
6. http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/
7. https://www.tensorflow.org/

5 Résultats

Pour obtenir une reproductibilité dans les expériences, la seed de tensorflow a été fixé arbitrairement à 12. Cela signifie qu'à chaque relance du programme, les paramètres seront initialisés de la même façon. Si la seed n'est pas fixée, d'une utilisation à l'autre le résultat varie, en bien ou en mal, et ne permet donc pas de tester plusieurs configurations en variant le bias, le weight, les features, etc.

Le modèle original obtient un score de 0.42 en utilisant les features tf-idf (n=20 tokens), un pas d'apprentissage à 0.0008, un nombre d'epochs de 27 000, le weight et le bias ont été initialisé selon une distribution gausienne avec une stddev fixée à environ 2.

À partir de cette baseline, nous avons fait varier plusieurs paramètres pour tenter une amélioration du modèle. Le meilleur résultat obtenu est 0.59, soit un gain de 0.19.

5.1 Variation des features

Tout d'abord, une première approche est l'ajout de features. On peut faire varier le nombre de tokens, n, renvoyé par le calcul du tf-idf, de base il est de 20. Voici une évolution des résultats selon n.

n	0	5	10	15	20	25	30	35
Score	/	**0.42**	0.38	0.40	**0.42**	0.26	0.34	0.36

TABLE 2 – Évolution de la précision selon le nombre de tokens utilisé

Le score le plus haut, 0.42, est obtenu avec n = 5 ou 20.

Si on rajoute les features personnalisées avec n fixé à 20, le score atteint 0.42 de précision. Une diminution se fait donc ressentir, néanmoins il est également possible de varier n, pour voir s'il existe une corréalation entre les features tf-idf et celles personnalisées.

n	0	5	10	15	20	25	30	35
Score	0.38	0.44	0.42	0.42	0.42	0.44	**0.53**	0.38

TABLE 3 – Évolution de la précision selon le nombre de tokens utilisé avec les features personnalisées

Ici, le meilleur score est obtenu avec n = 30, grâce à nos features personnalisées, la précision augmente de 0.11.

Les features sont indiquées par 0 ou 1, absence ou présence, dans le texte étudié. Il est à noté que si l'on retourne le nombre d'occurrences de chaque feature dans le texte, alors le modèle chute drastiquement.

Par la suite, nous ré-utiliseront les configurations donnant le meilleur score, ainsi dans la section suivante, n sera fixée à 30 et les features personnalisées seront utilisées.

5.2 Epochs

Les epochs correspondent au nombre de fois pendant lequel va ajuster ses poids. Si les epochs sont fixées à 10 000 alors le modèle va boucler 10 000 sur lui-même. Un taux d'epochs faible risque de ne pas permettre d'atteindre la précision maximale, alors qu'un taux trop élevé risque d'introduire du bruit car le modèle va sur-apprendre.

Epochs	1000	5000	10 000	15 000	20 000	25 000	26 000	27 000	35 000
Score	0.36	0.42	0.44	0.46	0.44	0.51	0.51	**0.53**	0.51

TABLE 4 – Précision selon le nombre d'epochs

Nous pouvons voir que la précision augmente avec le nombre d'epochs, obtenant le meilleur score à 27 000, à partir de 35 000, la précision diminue. Par défaut, notre nombre d'epochs produit donc le meilleur résultat.

5.3 Learning rate

Le learning rate est représenté par la fonction « tf.train.exponential_decay », grâce à elle, il est possible de varier plusieurs paramètres comme le pas, le global step et le decay rate. Nous allons donc modifier la valeur de chacun de ses paramètres afin de tenter une amélioration de la précision. Par défaut, le pas est fixé à 0.0008, le global step à 1 et le decay rate à 0.95.

5.3.1 Pas d'apprentissage

Pas	0.1	0.01	0.001	0.0008	0.0001	0.00001	1
Score	0.34	0.40	0.51	**0.53**	0.40	0.34	0.34

TABLE 5 – Précision selon le pas d'apprentissage

Le pas augmente progressivement à mesure que le pas d'apprentissage diminue, néanmoins s'il est trop faible, la précision se dégrade. Encore une fois, le taux par défaut donne les meilleurs résultats.

5.3.2 Global step

Global step	0	0.5	1	1.5	2	2.5
Score	**0.53**	0.38	**0.53**	0.38	**0.53**	0.38

TABLE 6 – Précision selon le global step

Tant que le global step est un chiffre rond, alors la précision ne varie pas, cependant on peut voir une dégradation dès lors que celui-ci est fixé à une valeur intermédiaire de 0.5. Nous garderons donc notre valeur de 1.

5.3.3 Decay rate

Decay rate	0	0.5	1	1.5	2	2.5
Score	0.53	0.53	0.53	0.53	0.53	0.53

TABLE 7 – Précision selon le decay rate

Comme le montre la table 7, le decay rate n'influe pas sur la précision, nous prendrons donc une valeur de 1 pour la suite.

5.4 Weight

Le poids accordé aux variables est très important, il est donc intéressant de tester plusieurs poids différents pour obtenir le modèle le plus performant. Comme nous avons fixé la seed à 12, la distribution gaussienne sera toujours initialisée de la même façon. Pour faire varier cette distribution, il est possible de modifier la moyenne et l'écart-type de la distribution. Par défaut la moyenne est fixée à 0 et l'écart-type à environ 2.04.

5.4.1 Écart-type

Écart-type	0	0.5	1	1.5	2	2.5
Score	0.55	**0.59**	0.55	0.51	0.53	0.51

TABLE 8 – Précision selon l'écart-type utilisée pour la distribution gaussienne pour initialiser le weight

Modifier la valeur de l'écart-type, nous permet d'améliorer le modèle, en effet d'après la table 8, pour un écart-type de 0.5, la précision augmente à 0.59.

5.4.2 Moyenne

Moyenne	0	0.5	1	1.5	2	2.5
Score	**0.59**	0.53	0.32	0.32	0.32	0.32

TABLE 9 – Précision selon la moyenne utilisée pour la distribution gaussienne pour initialiser le weight

La moyenne joue un rôle important, en effet on peut voir qu'à 0, elle permet d'obtenir le score maximal de 0.59, conjointement utilisée avec un écart-type de 0.5, néanmoins plus elle augmente, plus la précision décroît pour se fixer à 0.32.

5.5 Bias

Le bias permet de compenser la valeur attribuée à certains poids, il peut être intéressant de modifier sa valeur. Par défaut, le notre est initialisé comme pour le weight.

5.5.1 Écart-type

Écart-type	0	0.5	1	1.5	2	2.5
Score	0.55	0.55	0.57	0.57	**0.59**	**0.59**

TABLE 10 – Précision selon l'écart-type utilisée pour la distribution gaussienne pour initialiser le bias

L'augmentation de l'écart-type permet une augmentation du résultat, un plateau semble apparaître à partir de 2 comme le montre la table 10.

5.5.2 Moyenne

Moyenne	0	0.5	1	1.5	2	2.5
Score	**0.59**	0.57	0.57	0.55	0.55	0.55

TABLE 11 – Précision selon la moyenne utilisée pour la distribution gaussienne pour initialiser le bias

Encore une fois, la moyenne permet d'obtenir le score maximal lorsque celle-ci est à 0, plus elle augmente et plus la précision diminue (Table 11).

D'après les différentes variations du bias, il est à noter qu'aucune amélioration de la précision est apparue. Nos différentes features semblent donc assez équilibrées.

6 Discussion

Nous avons donc testé plusieurs configurations pour augmenter la précision de notre modèle consistant à prédire si un texte donné est fake, trusted ou parodic.

En partant d'une baseline à 0.42, nous sommes arrivés au final à obtenir une précision de 0.59. Pour se faire, les features utilisées ont joué un rôle important, l'introduction du score de tf-idf permet d'obtenir la baseline, et les features personnalisées ajoutent une amélioration. De plus, nous avons également fait varier les différentes valeurs pour l'entraînement du modèle comme le nombre d'epochs, le learning rate, le weight et le bias. Toutes ces modifications, nous ont permis d'obtenir la meilleure précision.

La matrice de confusion, représenté en figure 1, nous indique comment se comporte notre modèle lors de la classification. En observant la diagonale, on note que sur 16 articles fakes, notre système réussi à en annoter 12 correctement. Lorsque la catégorie est « trusted », notre système a plus de mal à bien classifier, en effet sur 16 articles, seulement 7 sont bien classifiés comme « trusted ». Il semble donc y avoir une confusion du système sur cette partie. Finalement, sur les 17 articles parodiques,

10 sont estimés correctement, néanmoins 6 sont également estimés comme « fake ». Il y a donc une confusion entre les articles fakes et parodiques.

Cette confusion entre les articles fakes et parodiques n'est pas vraiment étonnante, en effet les articles parodiques empruntent des éléments des articles sérieux, tout en introduisant des éléments faux. Il est donc plus difficile de distinguer ce genre de contenu humoristique.

Par ailleurs, il est intéressant de noter que malgré cette confusion, la distinction s'effectue plus précisément pour les catégories « fake » et « parodic ». En effet, la diagonale nous montre que l'identification de ces deux catégories est assez prononcée alors que pour la catégorie « trusted », la classification est plus difficile. On peut supposer que c'est justement lié au fait que les textes fakes et parodiques empruntent des éléments aux articles « trusted ». Certaines features très présentes dans un article fake ou parodique, se retrouvent donc également dans les articles sérieux, brouillant le modèle.

Pour conclure, nous avons donc une première confusion entre les articles parodiques et fakes qui ont un genre proche, puis une seconde confusion où l'on voit clairement que les articles sérieux sont plus difficiles à estimer de part leur lien commun avec les articles fakes et parodiques.

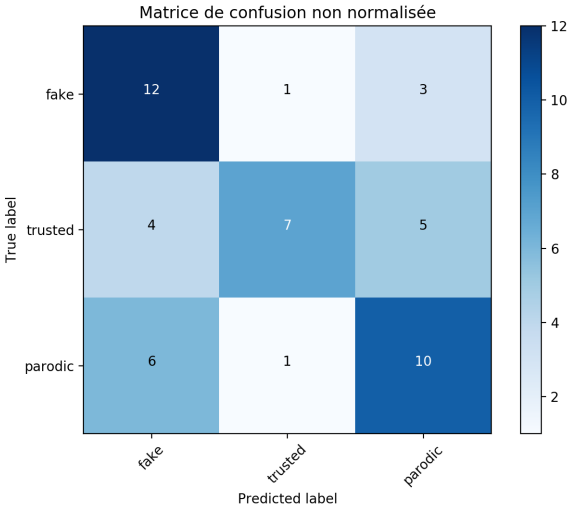

FIGURE 1 – Matrice de confusion non normalisée