Analyse de sentiments

Frédéric Wantiez – Pierre Vigier

1er juin 2016

1 Description du problème

1.1 Intérêt

1.2 Formalisation

L'analyse de sentiments est un problème d'apprentissage supervisé. Notons V l'ensemble de tous les mots possibles et $V^* = \cup_{n\geq 0} V^n$ l'ensemble des textes sur ce vocabulaire. Soit $x_1, ..., x_N \in V^*$ des textes et $y_1, ..., y_N \in S$ le sentiment associé à chacun des textes. Ces sentiments peuvent être des valeurs dans $S = [0, 1]$ où 0 signifie que le texte est "très négatif" et 1, "très positif". Une variante plus simple est d'avoir les $(y_i)_{i\in 1,...,N}$ dans $S = 0, 1$ où 0 signifie "négatif" et 1, "positif". Notre objectif est de déterminer une fonction f telle que $\forall i \in 1, ..., N, y_i \approx f(x_i)$ et qui devra, de plus, bien généraliser sur des textes jamais vus auparavant. Dans le cas où S est discret, il s'agit d'un problème de classification. Dans le cas continu, il s'agit d'un problème de régression.

Nous allons essentiellement nous concentrer sur le problème de classification. Plusieurs types d'entrée et plusieurs types de classifieurs seront essayés sur le problème. La mesure de performance choisie est la précision $A(y_1, ..., y_N, \hat{y}_1, ..., \hat{y}_N) = \frac{\sum_{i=0}^{N} 1_{y_i=\hat{y}_i}}{N}$. L'objectif est de la maximiser. Elle nous permettra de comparer les performances des différents algorithmes.

2 Données

Il est assez facile de créer un ensemble de données pour entraîner nos algorithmes. En effet, il suffit de trouver un site où l'on peut commenter et mettre des notes sur des produits. La valeur numérique de la note correspond alors au sentiment dégagé par le texte. Cette configuration est présente sur les sites d'e-commerce comme Amazon ou sur les sites de critiques comme IMDB ou Rotten Potatoes.

Nous utilisons l'ensemble de données mis à disposition par Maas et al. [1]. Il s'agit d'un ensemble de 50 000 avis en anglais sur IMDB. Il est découpé en un ensemble d'apprentissage et un ensemble de test. À chaque avis est associé un label 0 ou 1 selon que l'avis est positif ou négatif. Le label 0 correspond à une

note inférieure à 4 tandis que le label 1 correspond à une note supérieure à 7. Les notes de 5 et 6 sont exclues car représentant un avis neutre. Il y a autant d'avis positifs que d'avis négatifs dans l'ensemble d'apprentissage ainsi que dans l'ensemble de test. Dans la figure 1, on retrouve un avis négatif et un avis positif. On remarque que des fragments de code HTML et des libertés typographiques peuvent être présents.

id	sentiment	review
0_3	0	Story of a man who has unnatural feelings for a pig. Starts out with a opening scene that is a terrific example of absurd comedy. A formal orchestra audience is turned into an insane, violent mob by the crazy chantings of it's singers. Unfortunately it stays absurd the WHOLE time with no general narrative eventually making it just too off putting. Even those from the era should be turned off. The cryptic dialogue would make Shakespeare seem easy to a third grader. On a technical level it's better than you might think with some good cinematography by future great Vilmos Zsigmond. Future stars Sally Kirkland and Frederic Forrest can be seen briefly.
9999_8	1	The plot had some wretched, unbelievable twists. However, the chemistry between Mel Brooks and Leslie Ann Warren was excellent. The insight that she comes to, ""There are just moments,"" provides a philosophical handle by which anyone could pick up, and embrace, life. That was one of several moments that were wonderfully memorable.

FIGURE 1 – Exemple d'un commentaire négatif et d'un commentaire positif extrait de l'ensemble d'apprentissage.

2.1 Méthodologie

Afin de mener notre étude, nous avons choisi d'utiliser le langage Python (version 3.5) qui a l'avantage de présenter un écosystème de technologies liées à l'apprentissage automatique qui est développé et accessible. Nous utiliserons notamment les libraires suivantes :

— *pandas* pour manipuler et analyser des données ;
— *Numpy* pour l'algèbre linéaire ;
— *Matplotlib* pour les graphiques ;
— *scikit-learn* pour les algorithmes d'apprentissage automatique ;
— *TensorFlow* pour les réseaux de neurones ;
— *NLTK* pour manipuler le langage naturel.

Nous allons devoir faire face à plusieurs difficultés. Premièrement, nous devons prétraiter les avis en enlevant le code HTML. Nous devrons aussi réussir à modéliser les avis, qui sont des chaînes de caractères de taille variable, par des caractéristiques permettant d'utiliser des algorithmes d'apprentissage. La plupart d'entre eux nécessitant des entrées de taille fixe et numériques. Finalement, nous verrons que réussir à modéliser l'ordre des mots sera une nécessité pour atteindre améliorer la précision de la classification.

Notre démarche est assez simple, nous allons procéder par complexité croissante. Nous allons commencer par utiliser les modèles et algorithmes les plus "naïfs" puis analyser faiblesses de chaque méthode afin d'avancer et proposer des solutions pour améliorer nos résultats.

3 Premières tentatives

Nous allons commencer par utiliser deux représentations pour nos textes, la première étant les sacs de mots et la seconde se basant sur les vecteurs de mot. Dans chaque cas, nous allons nous concentrer sur la création des caractéristiques. Nous utiliserons pour commencer des algorithmes d'apprentissage relativement simples.

3.1 Sac de mots

3.1.1 Description

Les sacs de mots [4] est une représentation très simple. Numérotons les éléments de V, on a alors $V = w_1, ..., w_D$. Prenons un texte $t \in V^*$ et notons $tf_{i,t} = card(\{j, t_j = w_i\})$ le nombre d'occurrences du mot w_i dans le texte t. Le sac de mots d'un texte t est le vecteur b de $\mathbb{R}^D$ tel que $\forall i \in 1, ..., D, b_i = tf_{i,t}$. Autrement dit, la i^e coordonnée du sac de mots de t est le nombre d'occurrences du mot w_i dans t. Par exemple, si l'on considère les deux phrases suivantes :

$$S_1 = \text{"Bob aime les films d'action."}$$
$$S_2 = \text{"Alice préfère les films d'amour."}$$

Le vocabulaire commun aux deux phrases est :

$$V = \{\text{Bob}, \text{aime}, \text{les}, \text{films}, \text{d}, \text{action}, \text{Alice}, \text{préfère}, \text{amour}\}$$

En numérotant les mots dans l'ordre affiché ci-dessus, on a alors que les sacs de mots associés aux phrases S_1 et S_2 sont :

$$b_1 = (1, 1, 1, 1, 1, 1, 0, 0, 0)^T$$
$$b_2 = (0, 0, 1, 1, 1, 0, 1, 1, 1)^T$$

Afin d'obtenir des sacs de mots, nous avons nettoyé les critiques en enlevant le code HTML et en remplaçant la ponctuation par des espaces. Ensuite, nous avons utilisé les espaces pour découper les mots. Puis nous avons limité notre vocabulaire aux 5000 mots les plus courants afin d'avoir des vecteurs de taille raisonnable.

3.1.2 Apprentissage

Nous avons choisi de commencer nos tests en utilisant la régression logistique et les forêts aléatoires. Ces deux algorithmes ont l'avantage d'être facile à mettre en place et sont relativement rapide. De plus, ils sont relativement différents dans leur fonctionnement. La régression logistique modélise seulement une interaction linéaire entre les composantes du vecteur d'entrée contrairement aux forêts aléatoire. Cependant, on peut interpréter les coefficients de la régression logistique ce qui n'est pas possible pour les forêts aléatoires. On peut retrouver les différents résultats sur la figure 2.

Forêt aléatoire à 100 estimateurs + BOW	0.84356
Forêt aléatoire à 100 estimateurs + BOW + TF-IDF	0.83952
Régression logistique + BOW	0.84748
Régression logistique + BOW + TF-IDF	0.88308

FIGURE 2 – Précisions des sacs de mots avec différents algorithmes (BOW : sacs de mots, TF-IDF : sacs de mots pondérés par IDF)

Nous avons testé une variante des sacs de mots qui améliore significativement les résultats avec la régression logistique. Les composantes des sacs de mots sont maintenant pondérées par une mesure de la rareté d'un mot, IDF [4] que l'on définit comme suit :

$$\forall i \in \{1, ..., D\}, idf_i = log(\frac{N}{N_i})$$

où $N_i = cardk, tf_{i,x_k} > 0$ est le nombre de critiques qui contiennent le mot wi. La i_e composante du sac de mot d'un texte t est alors $b_i = tf_{i,t} \times idf_i$.

On peut retrouver sur la figure 3, les matrices de confusion pour la régression logistique et les forêts aléatoires. On remarque qu'il y a environ autant de faux positif que de faux négatifs.

La régression logistique a comme avantage qu'il est possible d'interpréter le modèle après apprentissage. Ainsi dans la régression logistique, à chaque mot est associé un poids. Et pour émettre une prédiction, le classifieur fait

		Sentiment prédit	
		0	1
Sentiment réel	0	10674	1826
	1	1987	10513

		Sentiment prédit	
		0	1
Sentiment réel	0	10692	1808
	1	2103	10397

FIGURE 3 – Matrice de confusion de la régression logistique et des forêts aléatoires pour des sacs de mots.

une combinaison linéaire des ces poids comme sur la figure 4. On peut alors interpréter le poids associé à chaque mot comme le sentiment que porte ce mot, seul. Dans les figures 5 et 6, on retrouve les mots jugés les plus négatifs et les plus négatifs par la régression logistique.

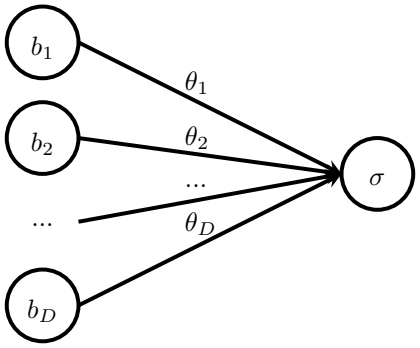


FIGURE 4 – Représentation d'une régression logistique sous forme de réseau.

3.2 Vecteurs de mots

3.2.1 Description

Une autre approche est d'essayer de représenter les mots avec une représentation continue. Le premier avantage de cette méthode est de fournir une représentation compacte des mots. En effet, avec les sacs de mots, les entrées sont des vecteurs de taille $card(V)$ ce qui induit un grand nombre de paramètres dans les algorithmes d'apprentissage ensuite. Au contraire, avec les représentations continus, nous pouvons fixer arbitrairement la taille des vecteurs qui vont représenter les mots du vocabulaire. Plus le vecteur aura de coordonnées, plus la représentation sera fine. Nous avons pas exemple choisi des vecteurs de $\mathbb{R}^{300}$. Nous noterons dans la suite v_i le vecteur de $\mathbb{R}^{300}$ associé au mot w_i de V.

Il existe de nombreuses méthodes pour générer de telles représentations, nous avons choisi le modèle Skip-gram proposé par Mikolov et al. [2] [3]. Le modèle consiste en un réseau de neurone, voir figure 7, où à partir d'un mot, nous essayons de prédire son contexte i.e. les mots qui l'entourent.

[h]

```
1. disappointment (-2.608113025883491)
2. waste (-2.359417022384978)
3. poorly (-2.3084246907186885)
4. baldwin (-2.19419686645024)
5. worst (-2.0717459858980107)
6. unwatchable (-2.0580492805825648)
7. obnoxious (-1.963499488620894)
8. lacks (-1.9554003844434584)
9. mst (-1.847654496703213)
10. forgettable (-1.8287902091116428)
```

FIGURE 5 – Les 10 mots ayant les poids associés les plus négatifs

[h]

```
1. refreshing (2.214710766134065)
2. vengeance (2.1735370232092186)
3. flawless (2.0122553306253668)
4. solo (1.7854059321112288)
5. voight (1.7756433401722442)
6. hooked (1.734909896519615)
7. wonderfully (1.7229447269614802)
8. existed (1.6963294241077245)
9. appreciated (1.5982250466066845)
10. stallone (1.5924422131212714)
```

FIGURE 6 – Les 10 mots ayant les poids associés les plus positifs

Ce modèle a l'avantage de fournir de l'information syntaxique et sémantique sur les mots. En effet, des mots qui se retrouvent souvent dans des contextes similaires auront des vecteurs proches. Et s'ils se retrouvent dans les contextes similaires c'est qu'ils ont des fonctions grammaticales et des sens proches. Pour s'en apercevoir, nous avons projeté nos vecteurs dans $\mathbb{R}^2$ pour les visualiser en utilisant l'algorithme t-SNE [5] qui préserve les voisinages (figure 3.2.1).

En particulier, on peut remarquer la présence de clusters de mots. En particulier, nous avons mis en évidence dans la figure 3.2.1 un cluster de mots liés au cinéma 3.2.1 et un autre contenant des mots-outils.

3.2.2 Apprentissage

Nous avons donc une représentation des mots cependant, nous avons besoin de représenter des textes. Une solution simple est de faire une combinaison linéaire des vecteurs représentants les mots du texte. Comme pour les sacs de

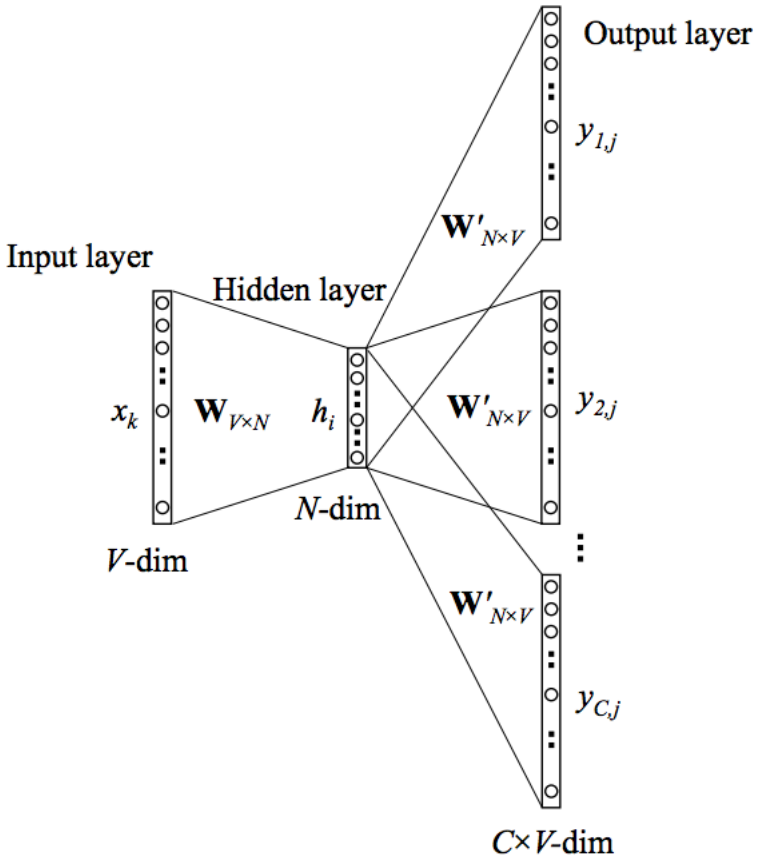


FIGURE 7 – Réseau de neurones du modèle Skip-gram (source : Wikimedia).

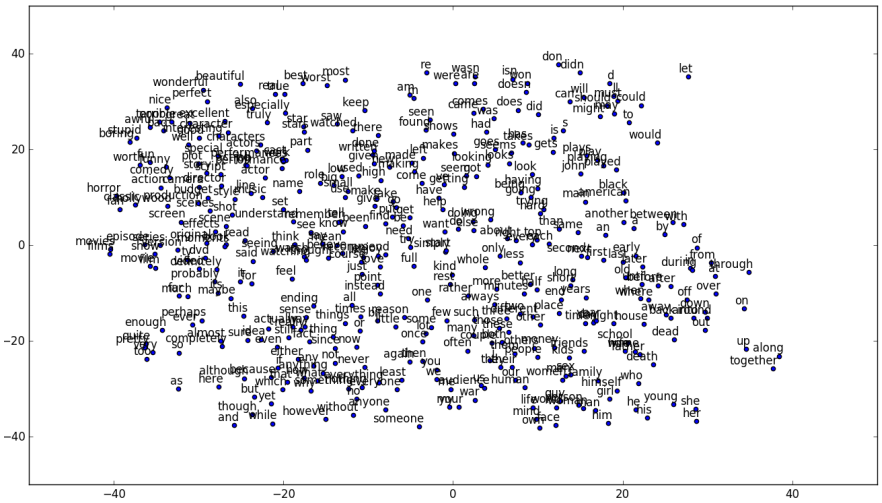
mots, plusieurs variantes s'offrent à nous. Nous pouvons supposer que tous les mots ont un poids égal, l'application associant à un texte t sont vecteur sera alors définie par :

$$h_{vec}(t) = \frac{\sum_{w_i \in t} tf_{i,t} v_i}{\sum_{w_i \in t} tf_{i,t}}$$

Cependant, on peut aussi considérer que les mots ont des poids différents selon leur rareté, et donc comme pour les sacs de mots, nous allons pondérer par l'IDF :

$$h_{vec+idf}(t) = \frac{\sum_{w_i \in t} tf_{i,t} idf_i v_i}{\sum_{w_i \in t} tf_{i,t} idf_i}$$

De même que pour les sacs de mots, nous avons mesurer la précision de la régression logistique et des forêts aléatoires avec comme entrées des vecteurs de



mot. On retrouvera ces résultats dans la figure 8.

Ici contrairement aux sacs de mots, la pondération des mots par l'IDF n'offre aucune amélioration. Cela peut s'expliquer par le fait que des mots rares mais ne portant aucun sens utile pour l'analyse de sentiment vont diminuer l'importance d'autres mots importants pour l'analyse de sentiment mais plus courants.

3.3 Conclusion

4 Prise en compte de l'ordre des mots

5 Conclusion

Références

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics : Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

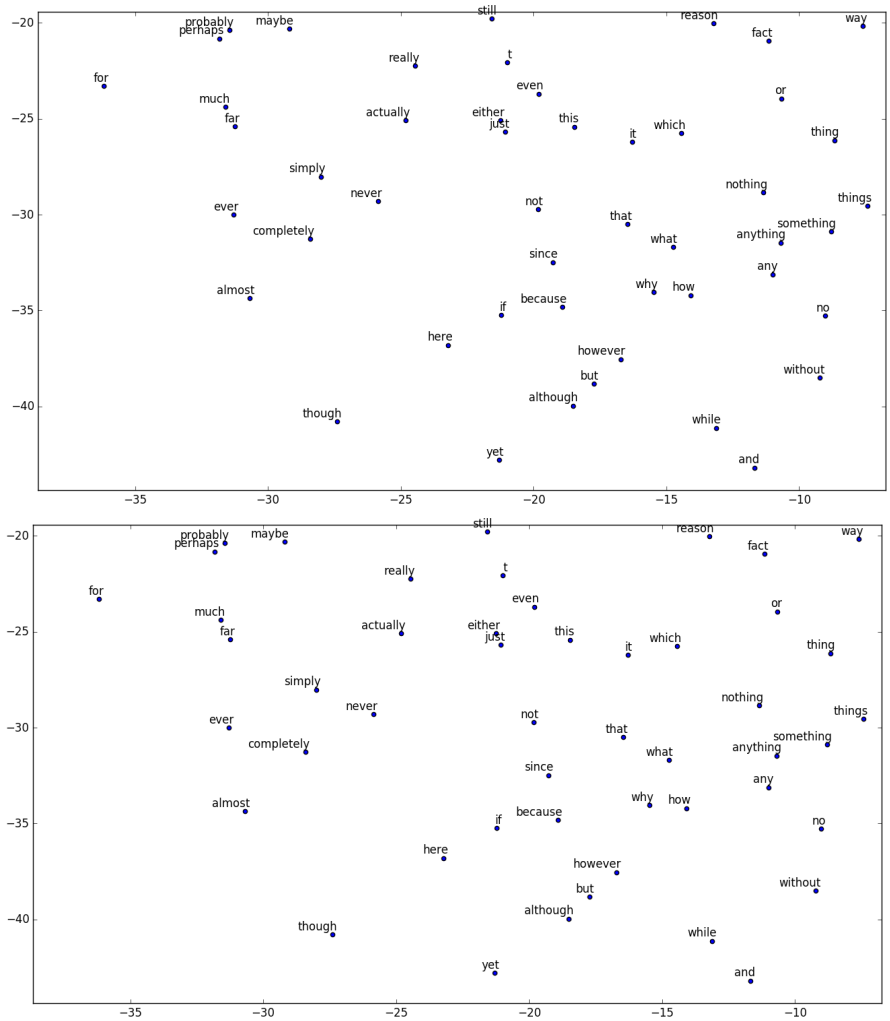
[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv :1301.3781*, 2013.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositiona-

lity. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[4] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[5] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605) :85, 2008.



Forêt aléatoire à 100 estimateurs + Vect	0.7954
Forêt aléatoire à 100 estimateurs + Vect + TF-IDF	0.79416
Régression logistique + Vect	0.81652
Régression logistique + Vect + TF-IDF	0.81688

FIGURE 8 – Précisions des vecteurs de mot avec différents algorithmes (Vec : vecteurs de mots, TF-IDF : vecteurs de mot pondérés par IDF)