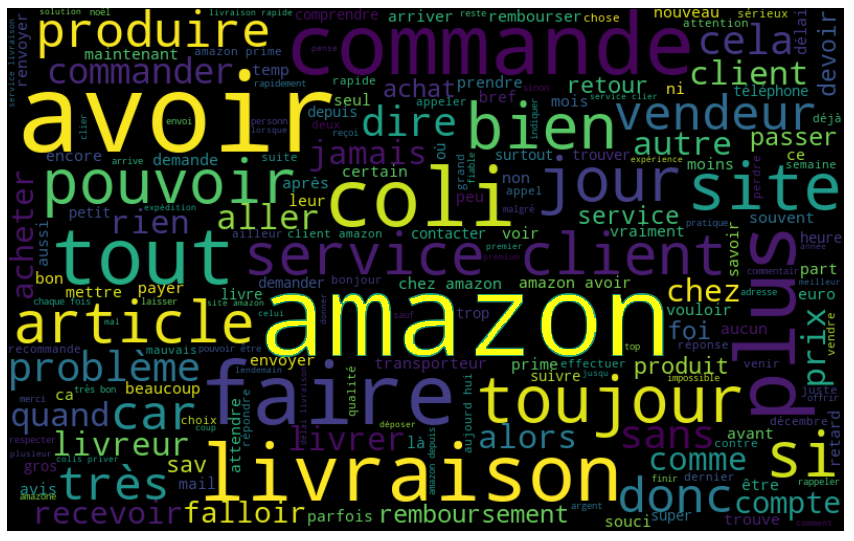
**Traitement du langage naturel**

****

**SAV des produits & services AMAZON**

**Table des Matières**

[I. Présentation du Sujet et Objectifs 2](#_Toc35127053)

[1. Biais sur l’analyse 3](#_Toc35127054)

[2. Pipeline des traitements 3](#_Toc35127055)

[II. Extraction des commentaires de Trustpilot 5](#_Toc35127056)

[III. Traitement des commentaires 9](#_Toc35127057)

[1. Prétraitement 9](#_Toc35127058)

[2. Tokénisation 10](#_Toc35127059)

[3. Suppression des Stop Words 11](#_Toc35127060)

[4. Lemmatisation 12](#_Toc35127061)

[5. Suppression des entités nommées 13](#_Toc35127062)

[6. Suppression des mots les plus fréquents 13](#_Toc35127063)

[IV. Représentation numérique des mots : TF-IDF 18](#_Toc35127064)

[V. Recherche d’un modèle de classification 20](#_Toc35127065)

[VI. Amélioration du modèle 23](#_Toc35127066)

[VII. ANNEXES 24](#_Toc35127067)

[1. Liste des stop words 24](#_Toc35127068)

[2. Liste des NER 25](#_Toc35127069)

[3. déNERification 26](#_Toc35127070)

# Présentation du Sujet et Objectifs

Le sujet de ce projet s’inscrit dans le domaine du traitement naturel du langage (TNL ou NLP[[1]](#footnote-1) en anglais).

Traiter le langage c'est :

* le comprendre pour le **traduire**, le **corriger**, le **résumer**,etc
* se contenter de le **classifier** sans l’avoir compris

La classification (analyse des sentiments) fait partie des principaux cas d'utilisation du TLN. Elle permet par exemple d'évaluer les commentaires publiés sur les réseaux sociaux pour voir comment se comporte la marque d'une entreprise.

Dans ce projet, pour la **classification**, nous avons fait face à des problèmes de nettoyage et d’analyse de données, d’apprentissage automatique et de prédiction. Le rapport explique les méthodes employées.

Concernant le sujet en lui-même, il a pour objectif de pouvoir assigner une note (1 à 5 étoiles) à un commentaire avec le maximum d’exactitude.

Ces commentaires existent réellement avec leur note. Ils proviennent du site [**Trustpilot**](https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=1&stars=2&stars=3&stars=4&stars=5)**,** plateforme d’avis où les clients commentent et notent les services et produits d’une société en particulier, en l’occurrence ici **Amazon**.

Le ‘scoring’ des commentaires est le suivant :

5\* : excellent

4\* : bien

3\* : moyen

2\* : bas

1\* : mauvais

Le **But** est donc de mettre au point un modèle capable d’assigner à un commentaire, la même note que lui a assigné le client.

## Biais sur l’analyse

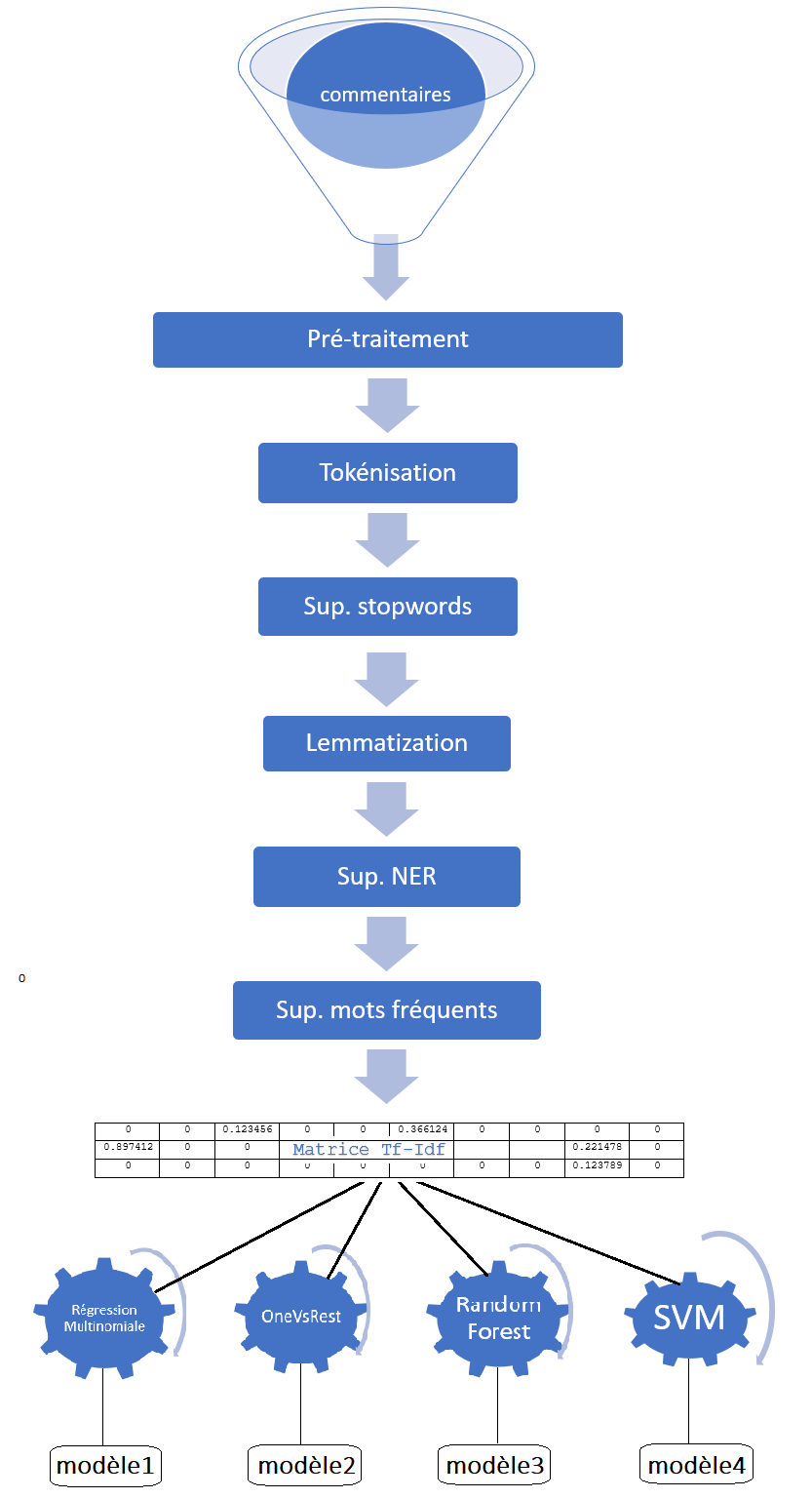
Malgré la tendance dégagée par l’analyse, la classification des commentaires est toutefois entachée d’un biais prévisible qu’il faut garder à l’esprit, comme par exemple :

* Lorsqu’un utilisateur attribue une note à son commentaire, il le fait de façon subjective ; à un problème identique, deux personnes peuvent noter différemment.
* Sur le site **Trustpilot,** il n’est pas nécessaire d’avoir un compte Amazon pour publier un commentaire sur les services Amazon. On aura donc forcément un % de Faux avis difficile à évaluer.
* La variété lexicale et le soin apporté à l’orthographe, est également propre à chaque personne. Un texte truffé de fautes d’orthographe perd de sa valeur.
* Certains commentaires sont trop courts.

## Pipeline des traitements

Le schéma suivant décrit dans l’ordre, l’ensemble des traitements appliqués aux commentaires puis aux mots jusqu’à l’obtention de quatre modèles de prédiction. L’idée est de rendre notre modèle plus adapté à notre problématique afin de donner de meilleures performances.

L’ensemble de ces petites actions pour modifier le corpus de mots, est la part humaine, non automatisable, qui demande un choix.



# Extraction des commentaires de **Trustpilot**

Sur le site web **Truspilot**, les commentaires 1, 2, 3, 4, 5\* sont respectivement accessibles via les adresses :

<https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=1>

<https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=2>

<https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=3>

<https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=4>

<https://fr.trustpilot.com/review/www.amazon.fr?languages=fr&stars=5>

A la date du 04/03/2020, il y a :

* 117 pages de commentaires 1\*
* 17 pages de commentaires 2\*
* 13 pages de commentaires 3\*
* 24 pages de commentaires 4\*
* 90 pages de commentaires 5\*

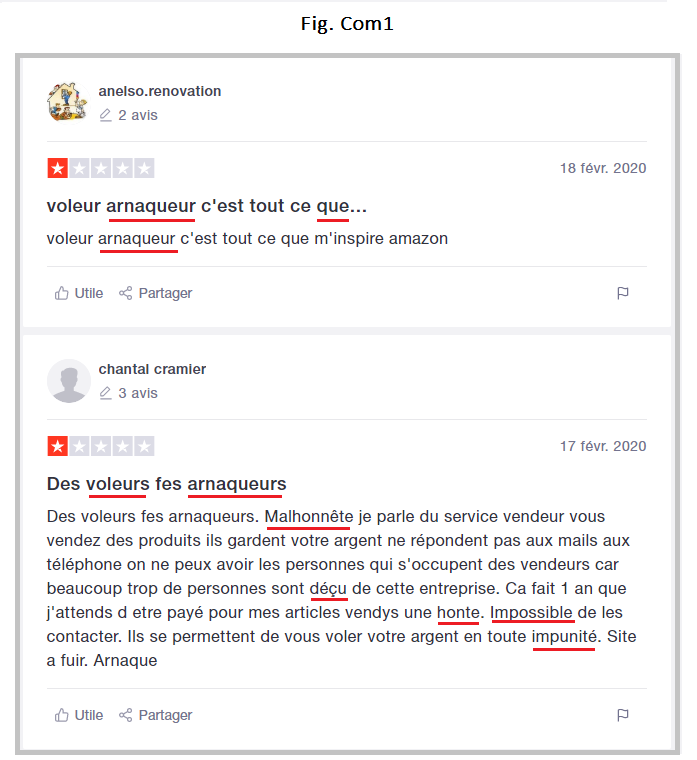
A propos des pages de commentaires : une page contient 20 commentaires.

L’extraction revient à identifier les 20 balises **review-card** ou sont insérés les commentaires (Fig. **review-card**).



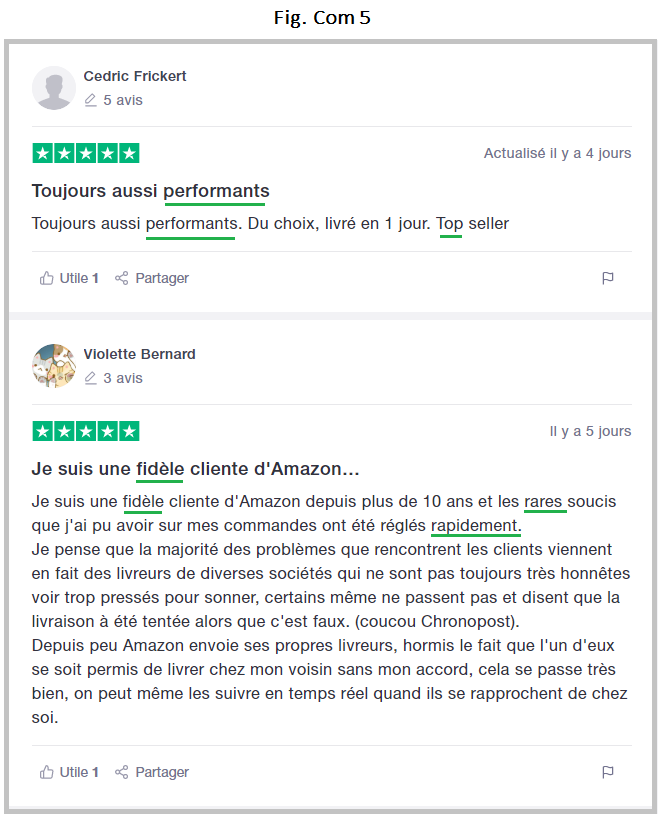
Voici deux exemples de commentaires extraits du site **Trustpilot**;

* un commentaire 1\* (voir Fig. **com1**)
* un commentaire 5\* (voir Fig. **com5**)



On remarque tout de suite une succession de termes négatifs\*.

C’est une hypothèse mais il est probable que la collecte de commentaire 1\* permettra de mettre en avant certains mots plus que d’autres, et surtout des mots que l’on rencontre beaucoup moins dans les autres catégories de commentaires.



Là aussi cet exemple affiche des termes positifs\* que l’on retrouvera, c’est une hypothèse, de préférence dans les commentaires 5\*.

# Traitement des commentaires

Quand on fait du traitement de documents, il y a tout un pipeline de traitements possibles. Certains sont inutiles. Tout dépend de la problématique.

Dans notre cas, on cherche à constituer un corpus de mots utilisés par catégorie de commentaire. On ne cherche pas à comprendre les **phrases**. C’est un tout autre traitement qui impliquerait d’avoir recourt au Deep Learning.

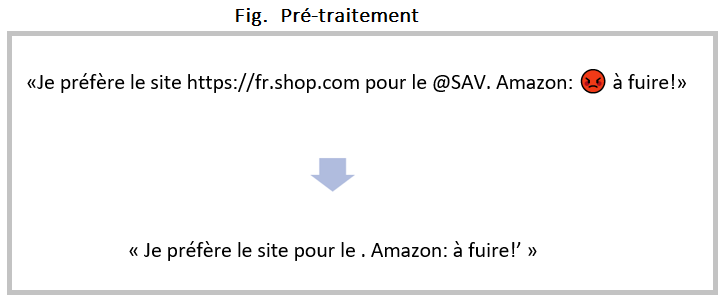
Ici on préfère se focaliser sur les **mots**. Certains mots ont de l’intérêt, d’autres pas. On va devoir les sélectionner et les traiter afin de rendre leur usage plus facile.

## Prétraitement

Certaines données ne présentent pas d’intérêt ou sont difficiles à interpréter comme les :

* liens hypertexte <https://fr.trustpilot.com/review/www.amazon.fr>
* Tags @Amazon, #SAV
* Emoji 😊😎😉😒

On préfère les supprimer (voir la fig. **Pré-traitement**)



Pour aller plus avant dans le nettoyage et le traitement des données, l’étape de ‘Tokénisation’ est indispensable.

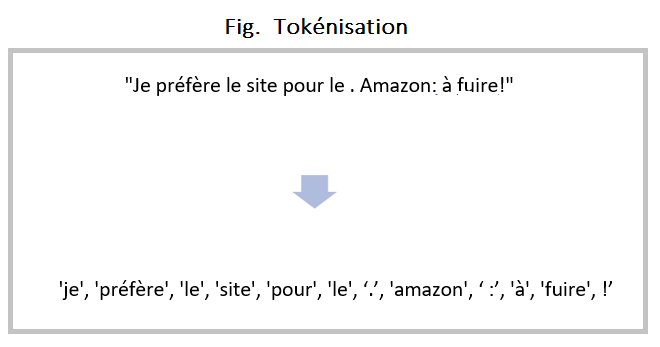
## Tokénisation

‘Tokéniser‘ revient à segmenter un texte de façon plus fine, en tokens (en). Ces tokens peuvent être des mots (unigrammes) ou ensembles de mots (n-grammes). La ponctuation, les symboles et les chiffres en font partie.

Nous choisissons de tokéniser les commentaires en unigrammes. En effet, notre projet ne consiste pas à modéliser le langage. On n’a pas besoin du contexte du mot (mots avant et après), seul le mot unitaire nous intéresse ici.

Certains mots composés perdront donc leur sens comme ‘**cerf-volant**’ qui deviendra ‘**cerf**’ + ‘**volant’**, mais ils seront rares par rapport à l’ensemble.

Exemple de commentaire (voir la fig. **Tokénisation**)

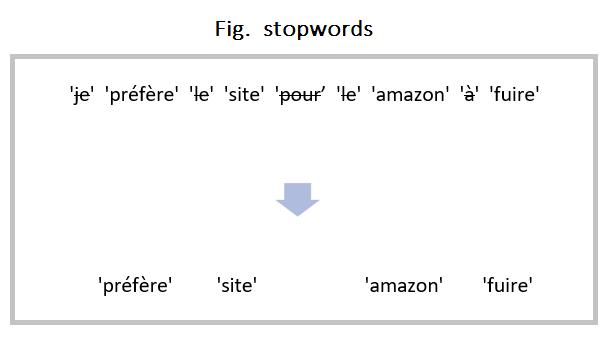


## Suppression des Stop Words

Les stop words sont les mots très courants de la langue étudiée ("et", "à", "le", ...) qui n’ont pas d’intérêt dans notre analyse. Leur nombre ralentit les traitements : nous les supprimons.

La liste des stop words est consultable en annexe.

Exemple de commentaire traité (voir fig. **stopwords**)



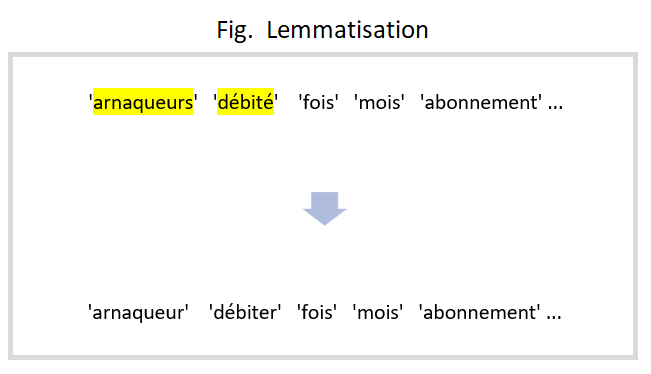
## Lemmatisation

La ‘lemmatisation’ consiste à représenter les mots (lemmes) sous leur forme canonique. Par exemple :

* Un verbe deviendra son infinitif
* Un nom deviendra son masculin singulier, idem pour les adjectifs

Ainsi le sens des mots utilisés dans le corpus est conservé et le nombre de mots à traiter diminue.

Exemple de commentaire lemmatisé (voir fig. **lemmatisation**)



## Suppression des entités nommées

La reconnaissance d’entités nommées (NER [[2]](#footnote-2)) permet de reconnaître les mots d’un texte qui correspondent à des concepts catégorisables (noms de personnes, lieux, organisations, ...).

Ici on se contente de retirer du corpus l'ensemble des ORG (organisation), PER (personne) et LOC (lieux).

La liste des NER détectés est consultable en annexe.

La détection des NER donne des résultats très aléatoires. Néanmoins leur suppression allège le volume du corpus à traiter par la suite et améliore notre modèle de prédiction.

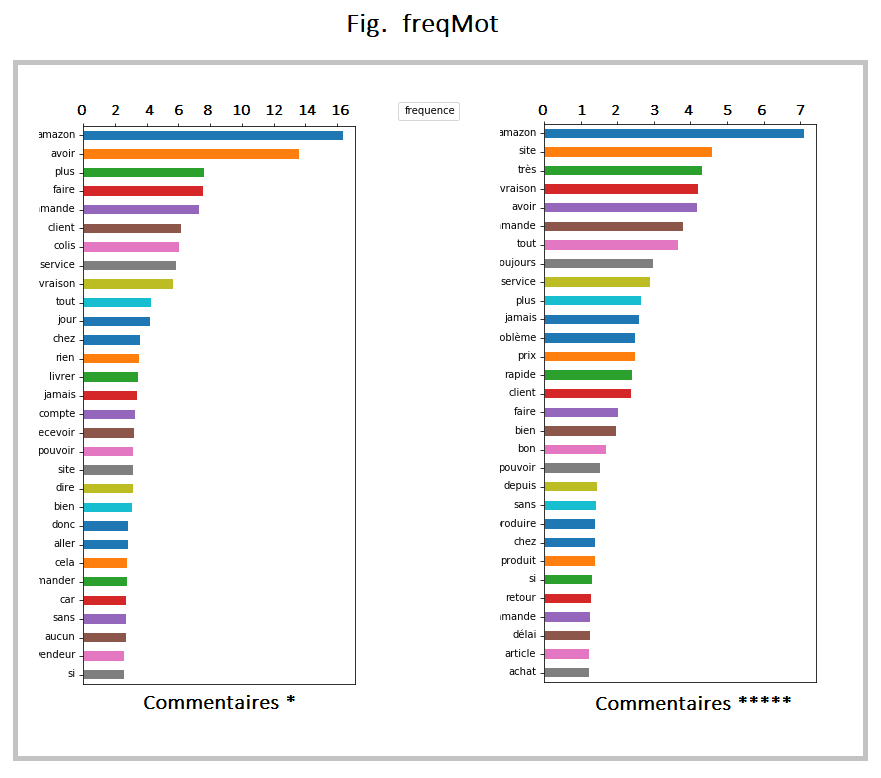
## Suppression des mots les plus fréquents

Pour cette étape l’idée est d’identifier les mots les plus fréquents par catégorie de commentaires (1\*, 2\*, 3\*, 4\*,5\*) afin de les supprimer.

Le nombre de mots à supprimer a été fixé à 30 après des tests.

Les graphiques suivants permettent de les visualiser sous deux formes :

* Statistique avec la figure **freqMot**
* Imagé avec la figure **wordcloud**



Cette figure représente les 30 mots les plus fréquents dans les catégories de commentaire **une** et **cinq** étoiles, avec la **Fréquence (F)** :

F = (nb apparitions du mot catégorie n / nb commentaires catégorie n) \*10

Difficile à première vue d’identifier les mots caractéristiques d’une catégorie de commentaires.

La figure **wordcloud** représente cette même fréquence sauf qu’elle ne tient pas compte du nombre de commentaires par catégorie. Elle a l’avantage de faire ressortir les mots à forte volumétrie.



Les mots comme ‘amazon’, ‘avoir’, ‘livraison’ dominent visuellement toutes catégories confondues.

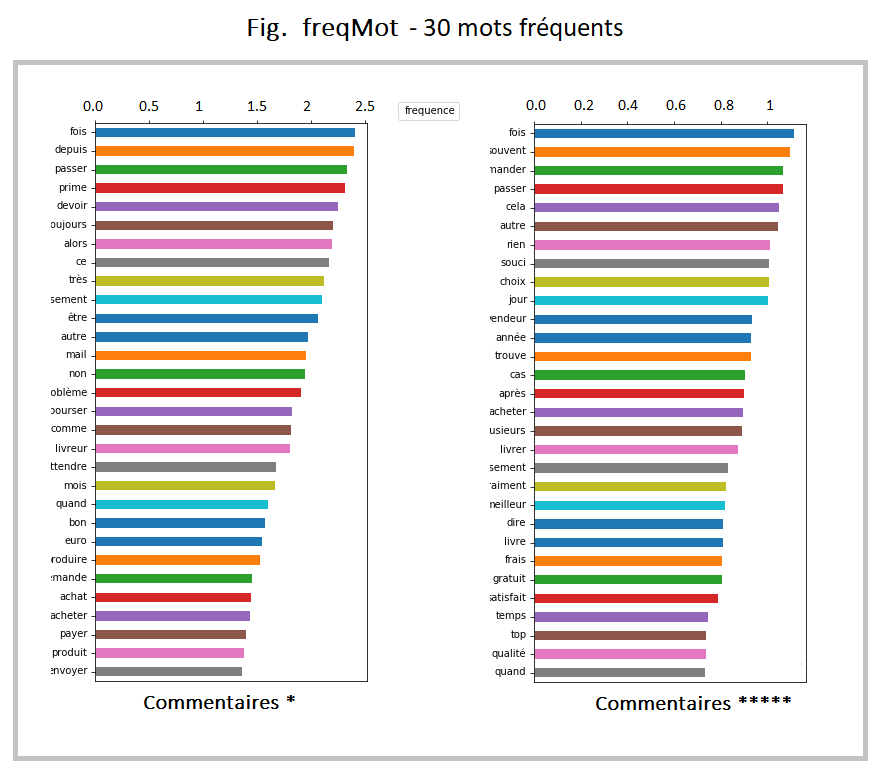
Des tests avec ou sans ces mots fréquents ont mis en évidence l’intérêt de les supprimer pour obtenir un meilleur modèle de prédiction.

Voir les mêmes figures après suppression des 30 mots :



Maintenant des différences apparaissent plus clairement d’une catégorie à l’autre.

Supprimer les 30 mots les plus fréquents a permis d’éliminer l’équivalent d’un **bruit** cachant les mots plus représentatifs des différentes catégories de commentaires.



Le mot ‘problème’ apparaît plutôt dans les commentaires 1\*.

Le mot ‘satisfait’ apparaît plutôt dans les commentaires 5\*.

# Représentation numérique des mots : TF-IDF

Après la préparation des données vient l’étape du traitement des données.

Comme un programme ne peut digérer du texte en entrée, les mots doivent être représentés sous une forme numérique qui ait un sens dans le contexte de notre projet.

Ici on a besoin de connaître le lexique de mots représentatifs de chaque catégorie de commentaires. On se doute bien que les mots se répartissent plus ou moins dans l’ensemble des commentaires. Par contre certains mots présentent un intérêt particulier car ils sont fréquents dans une catégorie de commentaires et rares dans les autres.

La métrique **TF-IDF** (Term-Frequency) – (Inverse Document Frequency) répond tout à fait cette notion de fréquence-rareté.

* **TF-IDF** = TF\*IDF
* TF : Nombre de fois que le mot apparaît dans un commentaire
* IDF : LOG (nb commentaires / nb commentaires contenant le terme)

Plus le TF-IDF est élevé, plus le mot est représentatif du commentaire.

Cette étape donne en sortie une **Matrice Tf-Idf**, une représentation numérique des mots ou chaque ligne représente un commentaire et chaque colonne représente un mot : une variable.

L’exemple du fichier Excel **features\_TfIdf** est une image de cette matrice. Une couleur est associée à chaque catégorie de commentaire. Les commentaires sont classés de 1 à 5\*.

Remarque : L’intérêt du tf-idf est aussi d’éliminer les mots qui n’ont pas pu être lemmatisés en raison d’une faute d’orthographe. Ils disparaissent ainsi.

Cette matrice Tf-Idf va servir à l’étape suivante ; l’entrainement des différents modèles.

C’est ici qu’intervient le Machine Learning. On a tendance à l’utiliser quand on ne peut pas résoudre quelque chose avec des règles car trop nombreuses dans le cas du langage. Le Machine Learning a besoin d’exemples, en grande quantité.

Dans notre cas, on entraîne nos modèles avec des commentaires labélisés, notés. L’ordinateur va trouver lui-même les règles de classification par apprentissage supervisé.

On donne donc à nos modèles algorithmiques, un ensemble de couples (**commentaire** + **note**) lui permettant d’apprendre à classifier un commentaire.

En réalité, on donne à notre modèle la matrice Tf-Idf ou les commentaires sont sous une forme vectorisée.

Si l’apprentissage s’est bien passé, on obtient alors un **Taux de bon classement** acceptable qui nous permettra d’utiliser le modèle à des fins de prédiction : trouver la note d’un commentaire.

! Le **Taux de bon classement** sera notre métrique pour évaluer les différents modèles.

Voyons maintenant les modèles algorithmiques utilisés pour calculer ce taux de bon classement.

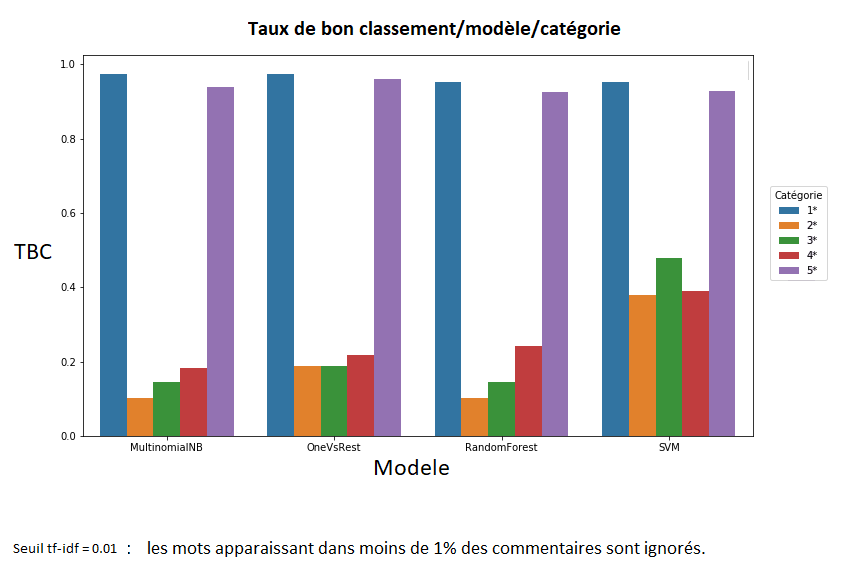
# Recherche d’un modèle de classification

Quatre modèles applicables à la classification de texte ont été testés :

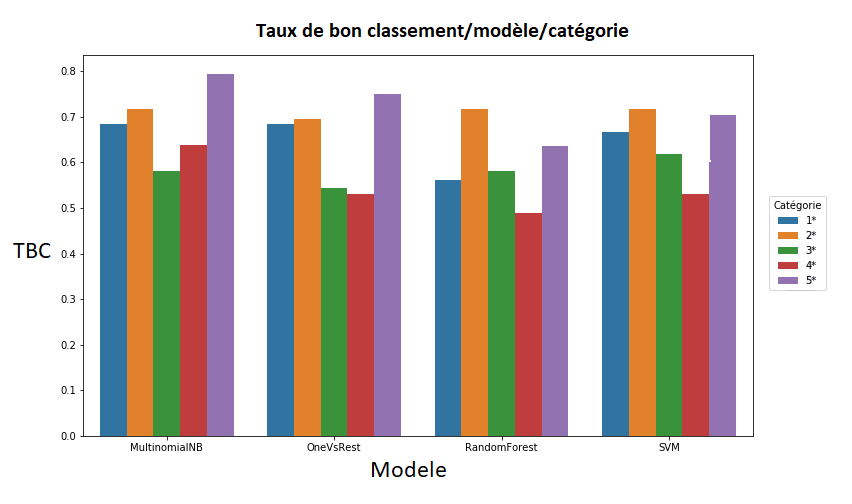
* Régression logistique multinomiale ([ref](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html).)
* OneVsRest ([ref](https://openclassrooms.com/fr/courses/4444646-entrainez-un-modele-predictif-lineaire/4507846-classifiez-vos-donnees-en-plus-de-deux-classes).)
* Random Forest ([ref](https://openclassrooms.com/fr/courses/4470521-modelisez-vos-donnees-avec-les-methodes-ensemblistes/4664688-reduisez-la-correlation-entre-les-apprenants-faibles-a-l-aide-des-forets-aleatoires).)
* Support Vector Machine ([ref](https://openclassrooms.com/fr/courses/4444646-entrainez-un-modele-predictif-lineaire/4507841-maximisez-la-marge-de-separation-entre-vos-classes).)

Ces quatre modèles nous donnent en sortie un **taux de bon classement** (tbc) en comparant les notes données à des commentaires tests par rapport à leur véritable note.

La figure suivante donne le **tbc** par modèle et par catégorie de commentaire :



La figure suivante donne aussi le **tbc,** mais cette fois en respectant l’égalité du nombre de commentaires par catégorie :



Analyse :

* Les tbc sont plus élevés lorsque le nombre de commentaires croit.
* Les tbc sont équivalents si le nombre de commentaires ayant servi d’entrainement est égal.
* Les 4 modèles donnent des résultats similaires.

Pour entraîner équitablement chaque catégorie de commentaires, il a fallu se ramener au plus petit nombre, celui des commentaires 3\*.

Mais le traitement du langage nécessite de gros volume de texte pour s’entraîner, d’où un tbc faible autour de 0.6.

Si on entraîne nos modèles avec la totalité des commentaires, on a certes un meilleur tbc moyen d’environ 0.8, mais avec une forte disparité selon les catégories.

Comment améliorer notre modèle ?

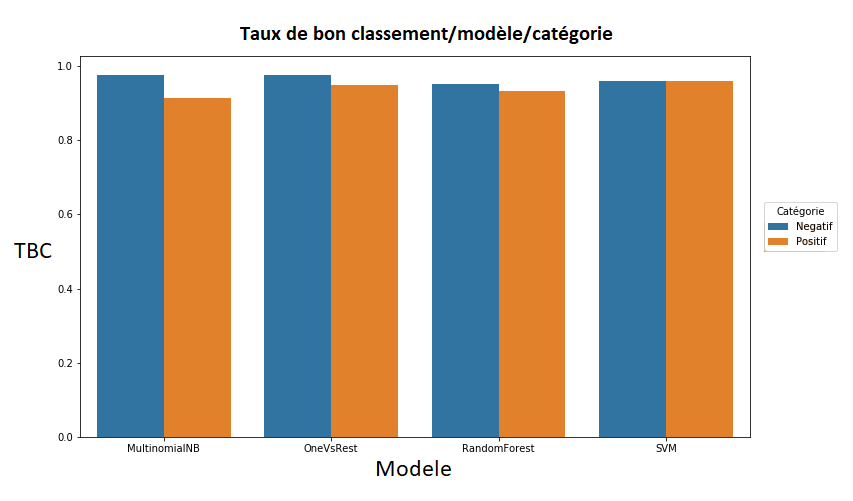
Nous pouvons le simplifier en réduisant le nombre de catégories :

* La catégorie 3\* est écartée car trop subjective.
* Les catégories 1,2\* forme la catégorie des commentaires négatifs
* Les catégories 4,5\* forme la catégorie des commentaires positifs

Réduire à deux catégories a plusieurs avantages :

* diminution de la subjectivité de la notation
* augmentation et égalisation du nombre de commentaires par catégorie

Voici les résultats dans la figure qui suit :



Le taux d’erreur d’environ 5% est satisfaisant et les tbc par catégorie ont des valeurs presque similaires.

Le modèle OneVsTest semble se détacher du lot, mais les autres modèles restent tout à fait applicable.

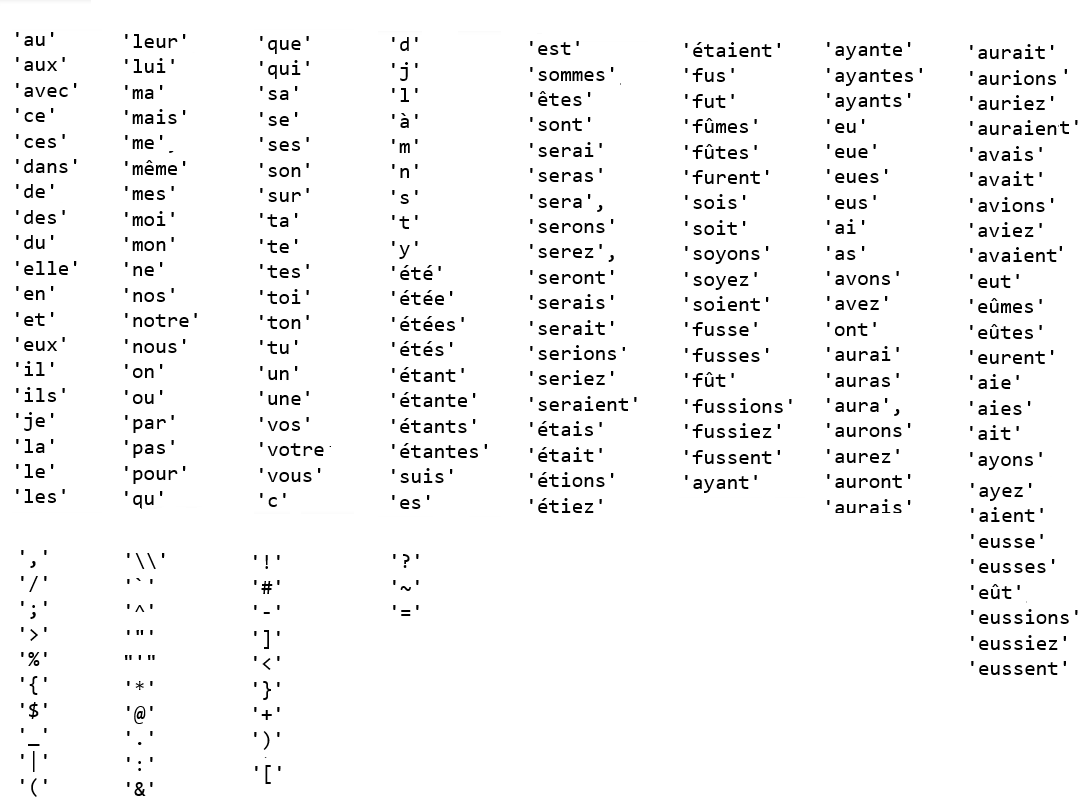
# Amélioration du modèle

En l’état notre modèle peut être amélioré avec :

* Une augmentation du nombre de commentaires
* Une égalisation du nombre de commentaires par catégorie
* Une changement pour un algorithme plus profond tel que le reseau de neurones
* Améliorer le traitement des Entités nommés
* Améliorer la lemmatisation
* Améliorer la suppression des stops words
* Proposer un correcteur orthographique

# ANNEXES

## Liste des stop words



## Liste des NER

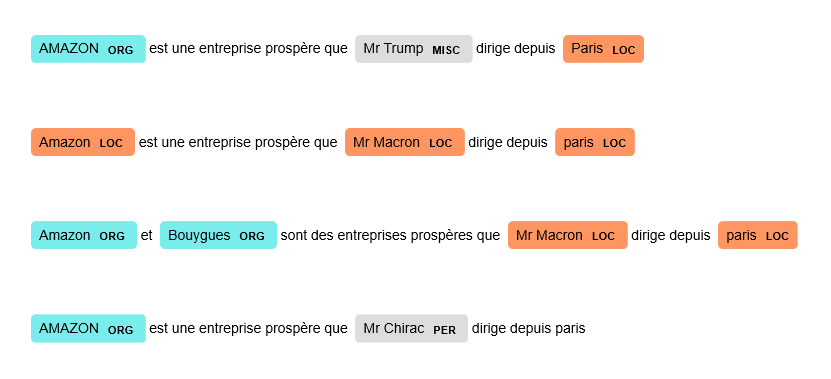
**Organisations** : ['ru', 'novembr', 'amazon', 'galèrer salopard', 'ps', 'marketplace amazon', 'colissimo international courrier post', 'eta', 'prb', 'amazon', 'inexister', 'samsung', 'samsung', 'europe', 'mèr', 'service fraud amazon', 'alexa amazon', 'amazon', 'amazon société arnaqueur', 'amazonvoilà', 'aid banque', 'cb', 'kms', 'amazon command', 'chronopost', 'amazon france', 'amazon logistics', 'amazon vendu', 'otr', 'customers first', 'amazon', 'cdiscount', 'hi fi', 'amazon', 'carte michelin', 'ds', 'amazon', 'cdiscount', 'marchandis amazon', 'livraison post', 'enorm', 'apré', 'cdiscount ebay', 'nike', 'kms avion', 'amazon', 'amazon colis arriver mauvais état', 'amazon fuyez sit', 'amazon logistics', 'maimouna amazon frmaimouna', 'maimouna', 'formulairemaimouner', 'cartemoi', 'apple', 'apple', 'amazon marchandise amazon', 'pub service', …

**Personnes** : ['amazone airpods', 'arnaqu', 'amazone moqu', 'amazone utilis', 'amazone ca', 'recevez marchandise', 'tondeuse oneblade', 'tondeus', 'clientèl nasreddine', 'apre raccroch', 'info', 'info', toutarticle', 'cest arnaqu', 'commandeet parcour', 'aupre', 'aupre 'chropost livraiser', 'blalalalala', 'problèm', 'commentair', 'jean louis', 'xiaomi', 'mensong', 'retir', 'certifié coli', 'desolé', 'iphon alia', 'harcèlemer', 'amazon', 'fuirmême aliexpress', 'demerdez problème', 'personn téléphon', 'imbecile deplorabl', 'esser', 'esser', 'fuyez misterlowcost marketplace', 'intervall désabonné', 'abo', 'discount', 'attentionaprès', 'remourser', 'amazon confiance', 'huawei pro', 'recontactee', 'renvoyez', 'sociètè crédit', 'articl', 'renvoyez bande', 'voulai gel nded', 'saint glin glin', …..

**Lieux** : ['occurrenc', 'bataill', 'maroc', 'mètr', 'cest', 'mentirune', 'cadr', 'ciao', 'nada', 'eur', 'france', 'dindon', 'pann', 'serrai', 'ca', 'hont', 'hont', 'noteamazon', 'cloturer', 'clier', 'france', 'pann', 'etat', 'france', 'rome', 'france', 'bonjouret', 'commandrai', 'collect', 'paris france', 'ca', 'france', 'france', 'clichy', 'novembr', 'chin', 'angleterre', 'amazone', 'cher', 'suisse', 'chin', 'amazonj', 'amazon', 'st priest', 'demand', 'coqu', 'incompetent', 'al tarmo', 'exigeon', 'hont', 'cheminéeil', 'articleune', 'boiscet', 'cobdulter', 'chine', 'st glinglin', 'france', 'rue gard', 'rue gard plain rue plain', 'ca', 'angleterre', 'france','cochon', 'paris', 'voila' …..

## déNERification

Supprimer les NER du corpus de mots s’appuie sur des bibliothèques qui ne donnent pas toujours les résultats attendus. En voici quelques exemples :



1. Natural langage Process [↑](#footnote-ref-1)
2. Named Entity Recognition [↑](#footnote-ref-2)