



ÉCOLE CENTRALE LYON

PROJET DE RECHERCHE
RAPPORT

NLP for sentiment analysis

Élèves :

Afaf EL KALAI

Enseignant :

Alexandre SAIDI

8 février 2022

Table des matières

| | | |
|----------|--|-----------|
| 1 | Introduction | 2 |
| 2 | Natural Language Processing | 2 |
| 2.1 | Preprocessing | 3 |
| 2.1.1 | Tokenisation | 3 |
| 2.1.2 | Suppression des mots vides (Stop words) | 4 |
| 2.1.3 | Stemming | 4 |
| 2.1.4 | Lemmatisation | 4 |
| 2.2 | Feature Engineering | 4 |
| 2.2.1 | Sac de mots | 5 |
| 2.2.2 | TF-IDF | 5 |
| 3 | Domaines d'application du NLP | 7 |
| 3.1 | Analyse des sentiments | 7 |
| 3.1.1 | Lexicon-Based Approach | 7 |
| 3.1.2 | Supervised Learning Approach | 9 |
| 3.2 | Topic Modeling | 15 |
| 3.2.1 | Latent Dirichlet Allocation | 16 |
| 4 | Data Mining : approche espace vectoriel | 20 |
| 4.1 | L'approche TF-IDF | 20 |
| 4.2 | Word Embedding : Word2vect | 20 |
| 4.2.1 | Continuous bag-of-words (CBOW) : | 20 |
| 4.2.2 | Skip-Gram Model : | 25 |
| 5 | Application : Analyse des Sentiments sur une base de données des Tweets | 26 |
| 5.1 | Introduction : | 26 |
| 5.2 | Base de Données utilisées | 27 |
| 5.3 | Échantillonnage des données non équilibrées | 27 |
| 5.4 | Représentation numérique du Dataset par la méthode TF-IDF | 28 |
| 5.5 | Réduction de la dimensionnalité : | 30 |
| 5.6 | Résultats | 30 |
| 5.6.1 | Méthode basée sur l'approche TF-IDF | 30 |
| 5.6.2 | Méthode basée sur l'approche word2vec | 32 |
| 6 | Conclusion | 34 |
| 7 | Remerciements | 34 |
| 8 | Bibliographie | 34 |

1 Introduction

Entre la création d'Internet et l'aube de 2003, les sites de médias sociaux comme Facebook, LinkedIn, MySpace transportaient quelques dizaines d'exaoctets d'informations sur le web alors que la même quantité est générée chaque semaine. En outre, la forme de communication la plus répandue sur le web existe sous forme de texte, ce qui offre une riche plate-forme pour exprimer ses émotions. En absence de contact direct, pour détecter les expressions faciales et les intonations de la voix, l'option alternative est de déchiffrer les sentiments à partir du texte dans les forums en ligne. L'analyse des sentiments à partir de documents textuels est un domaine populaire pour la recherche en traitement du langage naturel (NLP) et l'exploration de texte (Data Mining).

Les entreprises s'intéressent à ce que les clients disent de leurs produits. Les politiciens sont intéressés à savoir comment leur image est construite à travers les médias d'information. L'analyse des sentiments se concentre sur l'attribution d'une polarité ou d'une force à des expressions subjectives (mots et phrases qui expriment des opinions, des émotions, sentiments, etc.) afin de déterminer l'orientation objective/subjective d'un document ou la polarité positive/négative/neutre d'une phrase d'opinion dans un document. Même dans le cadre de la veille économique et des systèmes d'aide à la décision, l'analyse des sentiments est effectuée sur d'énormes volumes de données textuelles acquises à partir de diverses sources. De nombreux systèmes existent aujourd'hui pour accomplir la tâche d'analyse des sentiments.

2 Natural Language Processing

Le traitement automatique du langage naturel (Natural Language Processing NLP) est un sous-domaine de la linguistique, de l'informatique et de l'intelligence artificielle qui s'intéresse aux interactions entre les ordinateurs et le langage humain, en particulier à la manière de programmer les ordinateurs pour traiter et analyser de grandes quantités de données en langage naturel. Le résultat est un ordinateur capable de "comprendre" le contenu des documents, y compris les nuances contextuelles de la langue qu'ils contiennent. La technologie peut alors extraire avec précision les informations et les idées contenues dans les documents, ainsi que classer et organiser les documents eux-mêmes.

Afin de construire un modèle de Machine Learning capable d'amener une analyse de sentiment fiable un ensemble de tâches doivent être réalisées. Tout d'abord, les corpus de textes doivent être prétraités en se concentrant sur la réduction du vocabulaire et des distractions. Par distractions, j'entends les éléments qui empêchent l'algorithme (par exemple, les signes de ponctuation et la suppression des mots d'arrêt) de saisir les informations linguistiques essentielles à la tâche.

Viennent ensuite plusieurs étapes d'ingénierie des caractéristiques «Feature Engineering» : . L'objectif principal de l'ingénierie des caractéristiques est de faciliter l'apprentissage pour les algorithmes. Les caractéristiques sont souvent conçues à la main et orientées vers la compréhension humaine d'une langue. L'ingénierie des caractéristiques était d'une importance capitale pour les algorithmes classiques du langage naturel, et par conséquent, les systèmes les plus performants avaient souvent les caractéristiques les mieux conçues. Par exemple, pour une tâche de classification des sentiments, vous pouvez représenter une phrase par un arbre d'analyse et attribuer des étiquettes positives, négatives ou neutres

à chaque nœud/sous-arbre de l'arbre pour classer cette phrase comme positive ou négative. En outre, la phase d'ingénierie des caractéristiques peut utiliser des ressources externes telles que **WordNet** (une base de données lexicale) pour développer de meilleures caractéristiques. Nous allons bientôt examiner les différents techniques d'ingénierie des caractéristiques.

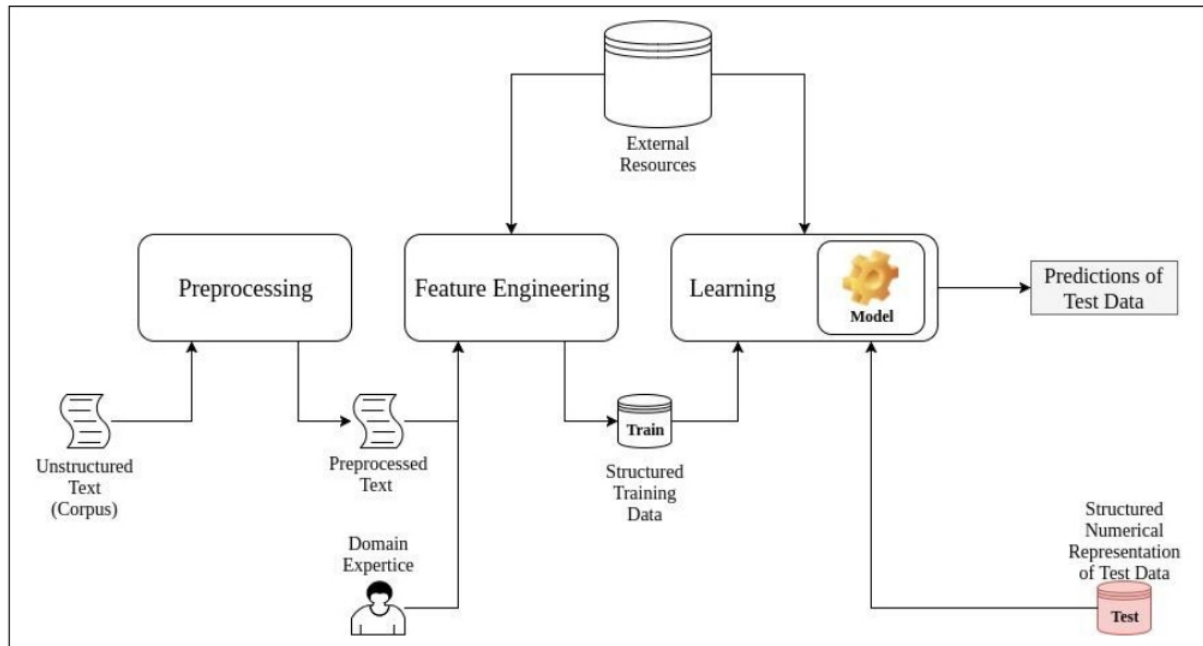


FIGURE 1 – L'approche classique du NLP

2.1 Preprocessing

2.1.1 Tokenisation

La tokenisation est souvent la première étape d'un modèle NLP. Un document est divisé en plusieurs parties pour en faciliter la manipulation. Souvent, chaque mot est un token, mais ce n'est pas toujours le cas, et la tokenisation doit savoir ne pas séparer les numéros de téléphone, les adresses e-mail, etc. Par exemple, voici les tokens pour la phrase d'exemple "Quand partirez-vous pour l'Angleterre?"

"Quand", "partirez", "vous", "pour", "l", "Angleterre"

Cet exemple de tokenisation semble simple car la phrase peut être divisée sur les espaces. Cependant, toutes les langues n'utilisent pas les mêmes règles pour diviser les mots. Pour de nombreuses langues d'Asie de l'Est, comme le chinois, la tokenisation est très difficile car aucun espace n'est utilisé entre les mots, et il est difficile de trouver où un mot se termine et où le mot suivant commence. L'allemand peut également être difficile à tokeniser en raison des mots composés qui peuvent être écrits séparément ou ensemble selon leur fonction dans une phrase.

La tokenisation n'est pas non plus efficace pour certains mots comme "New York". New et York peuvent avoir des significations différentes et l'utilisation d'un jeton peut donc prêter à confusion. C'est pourquoi la tokenisation est souvent suivie d'une étape

appelée "chunking", au cours de laquelle nous réunissons les expressions à plusieurs mots qui ont été séparées par un tokeniser.

La tokenisation peut être inadaptée au traitement des domaines textuels qui contiennent des parenthèses, des tirets et d'autres signes de ponctuation. La suppression de ces détails brouille les termes. Pour résoudre ces problèmes, les méthodes suivantes présentées ci-dessous sont utilisées en combinaison avec la tokenisation.

2.1.2 Suppression des mots vides (Stop words)

Après la tokenisation, il est courant d'éliminer les mots vides, c'est-à-dire les pronoms, les prépositions et les articles courants tels que "la" et "pour". En effet, ces mots ne contiennent souvent aucune information utile à nos fins et peuvent être supprimés sans risque. Cependant, les listes de mots vides doivent être choisies avec soin, car une liste qui convient à un objectif ou à un secteur d'activité peut ne pas être correcte pour un autre. Pour s'assurer qu'aucune information importante n'est exclue au cours du processus, un opérateur humain crée généralement la liste des mots vides.

2.1.3 Stemming

Stemming est le processus qui consiste à supprimer les affixes. Cela inclut les préfixes et les suffixes des mots. Des mots comme "merveilleusement" sont convertis en "merveilleux". Pour effectuer le "stemming", une liste commune d'affixes est créée, et ils sont supprimés de manière programmatique des mots en entrée. Le stemming doit être utilisé avec prudence car il peut modifier le sens du mot. Cependant, les stemmer sont faciles à utiliser et peuvent être modifiés très rapidement. Le « PorterStemmer » est un abrégiateur couramment utilisé en anglais et dans d'autres langues.

2.1.4 Lemmatisation

La lemmatisation a un objectif similaire à celui de la stemming : les différentes formes d'un mot sont converties en une seule forme de base. La différence est que la lemmatisation s'appuie sur une liste de dictionnaires. Ainsi, "ate", "eating" et "eaten" sont tous convertis en "eat", "going" et "went" sont convertis en "go" sur la base du dictionnaire, alors qu'un algorithme de lemmatisation ne serait pas en mesure de traiter cet exemple. Les algorithmes de lemmatisation ont idéalement besoin de connaître le contexte d'un mot dans une phrase, car la forme de base correcte peut dépendre de l'utilisation du mot comme nom ou comme verbe, par exemple. En outre, la désambiguïsation du sens des mots peut être nécessaire pour distinguer des mots identiques ayant des formes de base différentes.

2.2 Feature Engineering

Dans le monde réel, les données peuvent être extrêmement désordonnées et chaotiques. Peu importe qu'il s'agisse d'une base de données relationnelle SQL, d'un fichier Excel ou de toute autre source de données. Bien qu'elles soient généralement construites sous forme de tableaux où chaque ligne (appelée échantillon) possède ses propres valeurs correspondant à une colonne donnée (appelée caractéristique), les données peuvent être difficiles à comprendre et à traiter. Pour rendre la lecture des données plus facile pour nos modèles

d'apprentissage automatique et grâce à cela augmenter leurs performances, nous pouvons effectuer une ingénierie des caractéristiques.

2.2.1 Sac de mots

Il s'agit d'une technique d'ingénierie des fonctionnalités qui crée des représentations de caractéristiques basées sur la fréquence d'occurrence des mots. Par exemple, considérons les phrases suivantes :

- Ben went to the market to buy some flowers.
- Ben bought the flowers to give to Alice.

La liste vocabulaire pour ces deux phrases serait : ["Ben", "went", "to", "the", "market", "buy", "some", "flowers", "bought", "give", "Alice"]

Nous allons ensuite créer un vecteur (feature vector) de taille V (taille du vocabulaire) pour chaque phrase, indiquant combien de fois chaque mot de la liste vocabulaire apparaît dans la phrase. Dans cet exemple, les vecteurs de fonctionnalités pour les phrases seraient respectivement les suivants :

[1, 1, 2, 1, 1, 1, 1, 1, 0, 0, 0]
[1, 0, 2, 1, 0, 0, 0, 1, 1, 1, 1]

2.2.2 TF-IDF

Comme vous pouvez l'imaginer, l'approche Sac de mots pose un important problème de dimensionnalité : plus le nombre de documents est important, plus le vocabulaire est vaste, et la matrice des caractéristiques sera donc une énorme matrice éparsée. Par conséquent, le modèle de sac de mots est généralement précédé d'un important prétraitement (nettoyage des mots, suppression des mots vides, troncature/lemmatisation) visant à réduire le problème de dimensionnalité. La fréquence des termes n'est pas nécessairement la meilleure représentation du texte. En effet, on peut trouver dans le corpus des mots communs ayant la fréquence la plus élevée mais ayant peu de pouvoir prédictif sur la variable cible. Pour résoudre ce problème, il existe une variante avancée du Bag-of-Words qui, au lieu du simple comptage, utilise **la fréquence des termes - fréquence inverse des documents (ou Tf-Idf)**. Fondamentalement, la valeur d'un mot augmente proportionnellement au comptage, mais elle est inversement proportionnelle à la fréquence du mot dans le corpus.

La méthode TF-IDF est largement utilisée dans la recherche d'informations et l'exploration de textes. Le score TF-IDF du terme t dans le document d par rapport au corpus D est :

$$tfidf(t, d, D) = tf(t, d).idf(d, D)$$

Term Frequency (TF) Score

Nous devons d'abord calculer le $tf(t, d)$, qui est simplement le nombre de fois où chaque mot t est apparu dans le document d . Lors du calcul de $tf(t, d)$, nous éliminons généralement les stopwords. De plus, nous normalisons la fréquence des termes pour nous assurer qu'il n'y a pas de biais pour les documents plus longs ou plus courts. Ainsi, nous avons :

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$$

Avec $f_{t,d}$ représente le nombre d'occurrence du term t dans le document d .

Inverse Document Frequency (IDF)

Elle mesure la rareté d'un terme t dans l'ensemble du corpus D , c'est-à-dire la quantité d'informations qu'il fournit sur un document dans lequel il apparaît. Si le nombre total de documents dans le corpus est N , et n_t est le nombre de documents où t figurent, alors on a :

$$idf(t, D) = \log\left(\frac{N}{n_t}\right)$$

La raison pour laquelle nous prenons le log d'IDF est que si nous avons un grand corpus, les valeurs d'IDF deviendront si grandes, donc, nous utilisons la valeur log pour diminuer cet effet.

$f_{t,d}$

| | blue | bright | can | see | shining | sky | sun | today |
|---|------|--------|-----|-----|---------|-----|-----|-------|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 | 2 | 0 |

→

$tf(t, d) = \frac{f_{t,d}}{\sum_{t'} f_{t',d}}$

| | blue | bright | can | see | shining | sky | sun | today |
|---|------|--------|-----|-----|---------|-----|-----|-------|
| 1 | 1/2 | 0 | 0 | 0 | 0 | 1/2 | 0 | 0 |
| 2 | 0 | 1/3 | 0 | 0 | 0 | 0 | 1/3 | 1/3 |
| 3 | 0 | 1/3 | 0 | 0 | 0 | 1/3 | 1/3 | 0 |
| 4 | 0 | 1/6 | 1/6 | 1/6 | 1/6 | 0 | 1/3 | 0 |

$f_{t,d}$

| | blue | bright | can | see | shining | sky | sun | today |
|-------|------|--------|-----|-----|---------|-----|-----|-------|
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 3 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 1 | 1 | 1 | 0 | 2 | 0 |
| n_t | 1 | 3 | 1 | 1 | 1 | 2 | 3 | 1 |

→

$idf(t, D) = \log_{10} \frac{N}{n_t}$

| | blue | bright | can | see | shining | sky | sun | today |
|---|-------|--------|-------|-------|---------|-------|-------|-------|
| 1 | 0.602 | 0.125 | 0.602 | 0.602 | 0.602 | 0.301 | 0.125 | 0.602 |

$N = 4$

$\log_{10} \frac{4}{1} = 0.602$

$\log_{10} \frac{4}{3} = 0.125$

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

| | blue | bright | can | see | shining | sky | sun | today |
|---|--------------|--------|--------------|--------------|--------------|--------------|--------|--------------|
| 1 | 0.301 | 0 | 0 | 0 | 0 | 0.151 | 0 | 0 |
| 2 | 0 | 0.0417 | 0 | 0 | 0 | 0 | 0.0417 | 0.201 |
| 3 | 0 | 0.0417 | 0 | 0 | 0 | 0.100 | 0.0417 | 0 |
| 4 | 0 | 0.0209 | 0.100 | 0.100 | 0.100 | 0 | 0.0417 | 0 |

Une fois le texte d'entrée est transformé en une liste de valeurs numériques prêt à être interpréter par la machine, il est temps pour alimenter le modèle avec le jeu d'entraînement(structured training data).

3 Domaines d'application du NLP

3.1 Analyse des sentiments

Dans la littérature, il existe de nombreuses méthodes et algorithmes pour mettre en œuvre des systèmes d'analyse des sentiments, que l'on peut classer comme suit :

- Approche automatique : systèmes qui s'appuient sur des techniques d'apprentissage automatique à partir de données.
- Lexicon-based approche : systèmes qui utilise un lexique de sentiments préparé à l'avance pour évaluer un document en agrégeant les scores de sentiments de tous les mots du document étudié.

3.1.1 Lexicon-Based Approach

L'application d'un lexique est l'une des deux principales approches de l'analyse des sentiments. Elle consiste à calculer le sentiment à partir de l'orientation sémantique des mots ou des phrases qui apparaissent dans un texte. Avec cette approche, un dictionnaire de mots positifs et négatifs est nécessaire, avec une valeur de sentiment positive ou négative attribuée à chacun des mots. Différentes approches de création de dictionnaires ont été proposées, notamment des approches manuelles et automatiques. En général, dans les approches basées sur le lexique, un morceau de message textuel est représenté comme un sac de mots. Après cette représentation du message, des valeurs de sentiment provenant du dictionnaire sont attribuées à tous les mots ou phrases positifs et négatifs du message. Une fonction de combinaison, telle que la somme ou la moyenne, est appliquée afin d'effectuer la prédiction finale concernant le sentiment global du message. Outre la valeur du sentiment, l'aspect du contexte local d'un mot est généralement pris en considération, comme la négation ou l'intensification.

Les différentes approches de Lexicon-Based Approach sont :

3.1.1.1 Approche basée sur le dictionnaire

Dans cette approche, un dictionnaire est créé en prenant initialement quelques mots. Ensuite, un dictionnaire en ligne, un thésaurus ou WordNet peut être utilisé pour étendre ce dictionnaire en incorporant les synonymes et les antonymes de ces mots. Le dictionnaire est étendu jusqu'à ce qu'aucun nouveau mot ne puisse y être ajouté. Le dictionnaire peut être affiné par une inspection manuelle.

3.1.1.2 Approche basée sur le corpus

Appelée aussi méthode statistique, contrairement à la méthode précédente, l'analyse automatique des sentiments n'a pas besoin d'un dictionnaire de mots négatifs et positifs,

elle a plutôt besoin de deux corpus commentés (peut-être un seul, si on utilise l'apprentissage non supervisé), le premier corpus est dédié à l'apprentissage (training), à partir des commentaires, le modèle sera automatiquement capable de faire une analyse équivalente et de façon autonome. Le deuxième corpus est dédié au test, son intérêt est de valider la performance du modèle, si le cas est parfait, le résultat de l'analyse obtenue devrait correspondre à la totalité de celui du premier corpus. Pour une meilleure performance du modèle, il est important que le corpus de formation soit suffisamment représentatif pour le corpus d'évaluation .

3.1.1.3 Problèmes avec Lexicon-based approach

Il n'est pas surprenant que les indicateurs les plus importants des sentiments soient les mots de sentiment, également appelés mots d'opinion. Par exemple, "bon", "merveilleux" et "étonnant" sont des mots de sentiment positifs, tandis que "mauvais", "pauvre" et "terrible" sont des mots de sentiment négatifs. Outre les mots individuels, il existe également des phrases et des expressions idiomatiques, par exemple, "coûte un bras et une jambe". Les mots et expressions de sentiment sont essentiels dans l'analyse des sentiments. Une liste de ces mots et expressions est appelée lexique des sentiments (ou lexique d'opinion). Au fil des ans, les chercheurs ont conçu de nombreux algorithmes pour compiler de tels lexiques. Bien que les mots et phrases de sentiment soient importants, ils sont loin d'être suffisants pour une analyse précise des sentiments. Le problème est beaucoup plus complexe. Nous soulignons plusieurs problèmes dans ce qui suit :

1. Un mot de sentiment positif ou négatif peut avoir des orientations ou des polarités opposées dans différents domaines d'application ou contextes de phrases. Par orientation ou polarité, nous entendons le caractère positif, négatif ou neutre d'un sentiment ou d'une opinion. Par exemple, "suck" indique généralement un sentiment négatif, par exemple, "This camera sucks", mais il peut également impliquer un sentiment positif, par exemple, "This vacuum cleaner really sucks.". Ainsi, nous disons que les orientations des mots de sentiment peuvent dépendre du domaine ou même du contexte de la phrase.
2. Une phrase contenant des mots de sentiment peut ne pas exprimer de sentiment. Ce phénomène se produit dans plusieurs types de phrases. Les phrases de question (interrogatives) et les phrases conditionnelles sont deux types principaux, par exemple, "Pouvez-vous me dire quel appareil photo Sony est bon ?" et "Si je trouve un bon appareil photo dans le magasin, je l'achèterai. Ces deux phrases contiennent le mot sentimental bon, mais aucune n'exprime une opinion positive ou négative sur un appareil photo spécifique. Cependant, cela ne veut pas dire pas que toutes les phrases conditionnelles et interrogatives n'expriment aucune opinion ou aucun sentiment, par exemple : "Quelqu'un sait comment réparer cette terrible imprimante ?" et "Si vous cherchez une bonne voiture, achetez une Ford Focus."
3. Les phrases sarcastiques avec ou sans mots de sentiment sont difficiles à gérer, par exemple : "Quelle belle voiture! Elle a cessé de fonctionner en deux jours". Le sarcasme n'est pas si courant dans les avis des consommateurs sur les produits et services, mais il est fréquent dans les discussions politiques, ce qui rend les opinions politiques difficiles à gérer.

4. De nombreuses phrases sans mots de sentiment peuvent impliquer des sentiments ou des opinions positives ou négatives de leurs auteurs. Par exemple, "Ce lave-linge consomme beaucoup d'eau" implique une opinion négative sur le lave-linge car il consomme beaucoup de ressources (eau). Beaucoup de ces phrases sont en fait des phrases objectives qui expriment des informations factuelles. Par exemple, "Après avoir dormi sur le matelas pendant deux jours, une vallée s'est formée au milieu" exprime une opinion négative sur la qualité du matelas. Cette phrase peut être considérée comme objective car elle énonce un fait, bien que la vallée soit ici utilisée comme une métaphore. Comme nous pouvons le constater, ces deux phrases ne contiennent aucun mot de sentiment, mais elles expriment toutes deux quelque chose d'indésirable, ce qui indique des opinions négatives.

Toutes ces questions présentent des défis majeurs. En fait, ce ne sont que des exemples des problèmes difficile qui peuvent se poser dans un problème d'analyse des sentiments

3.1.2 Supervised Learning Approach

Les approches automatiques reposent sur des techniques d'apprentissage automatique (Machine Learning). La tâche d'analyse des sentiments est généralement modélisée comme un problème de classification dans lequel un classificateur est alimenté avec un texte et renvoie la catégorie correspondante, par ex. positif, négatif ou neutre (en cas d'analyse de polarité). On doit donc s'appuyer sur des techniques de machine Learning, et plus précisément d'apprentissage supervisé (en opposition avec des techniques d'apprentissage non-supervisé). Cela se résume à identifier des patrons dans des textes existants, où le sentiment global est connu, et d'appliquer ensuite ces patrons à de nouveaux documents, dont le sentiment est inconnu. On obtiendrait ainsi ce qu'on appelle un 'classificateur'. Parmi les techniques courantes employées, on peut citer :

- Arbre de décision : Un modèle qui permet de prédire la classe ou la valeur de la variable cible en apprenant des règles de décision simples déduites de données antérieures (données d'apprentissage).
- Naïve Bayes (NB) : Un modèle probabiliste simple utilisant le théorème de Bayes avec une hypothèse d'indépendance des variables aléatoires.
- Support Vector Machines (SVM) : un modèle qui permet de séparer l'espace de données en deux parties par un hyperplan.
- Extreme Learning Machine : L'ELM utilise une architecture de réseau neuronal de type feed-forward et fonctionne avec des poids d'entrée déterminés de manière aléatoire.

3.1.2.1 Arbre de décision

Le classificateur à arbre de décision fournit une décomposition hiérarchique de l'espace de données d'apprentissage dans laquelle une condition sur la valeur de l'attribut est utilisée pour diviser les données. La condition ou le prédicat est la présence ou l'absence d'un ou plusieurs mots. La division de l'espace de données est effectuée de manière récursive jusqu'à ce que les nœuds feuilles contiennent certains nombres minimums d'enregistrements qui sont utilisés pour le but de la classification.

Cependant, les arbres de décision sont souvent relativement imprécis. Pour remédier à ce problème, une nouvelle technique a été introduite : Random Forest. L'approche Random Forest est un algorithme d'apprentissage supervisé. Il est construit de multiples arbres de

décision, connus sous le nom de forêt, et les assemble pour obtenir une prédiction plus précise et plus stable.

La forêt aléatoire prend des échantillons aléatoires ou des observations, des variables initiales aléatoires (colonnes) et essaie de construire un modèle. L'algorithme de la forêt aléatoire est le suivant :

1. Tirer un échantillon aléatoire de taille n (choisir aléatoirement n échantillons parmi les données de formation).
2. Faire croître un arbre de décision à partir de l'échantillon. À chaque nœud de l'arbre, sélectionner aléatoirement d'autres caractéristiques.
3. Diviser le nœud en utilisant les caractéristiques (variables) qui fournissent la meilleure division selon la fonction objective. Par exemple, en maximisant le gain d'information.
4. Répétez les étapes 1 à 2, k fois (k est le nombre d'arbres que vous voulez créer en utilisant un sous-ensemble d'échantillons).
5. Agréger la prédiction de chaque arbre pour un nouveau point de données afin d'attribuer l'étiquette de classe par vote majoritaire, c'est-à-dire choisir le groupe sélectionné par le plus grand nombre d'arbres et attribuer le nouveau point de données à ce groupe.

3.1.2.2 Support Vector Machine

Le principe du SVM est de trouver un séparateur, donc un hyperplan (espace de dimensions $p - 1$, une droite pour le cas $p = 2$) de la forme $w^T \cdot x + b = 0$ qui permet de séparer les données d'entraînement en deux catégories, selon leur label. Avec $w = (w_1, \dots, w_p)^T$ un vecteur de poids constant et b un paramètre.

Dans un problème de classification par un modèle SVM, on peut tomber sur deux types de problèmes. Un problème linéairement séparable ou un problème linéairement inséparable. Un problème est linéairement séparable s'il existe un séparateur linéaire qui résout totalement le problème. S'il n'existe pas d'hyperplan qui permet de complètement classer les données en deux parties selon leur label alors on est devant un problème linéairement inséparable.

Considérant la fonction de décision $h(x) = w^T \cdot x + b$. Pour un x arbitraire, si $h(x) > 0$ alors l'entrée avec les données x correspondra à un label $y_i = +1$, et si $h(x) < 0$ alors elle correspondra à un label $y_i = -1$. Ainsi on peut définir la fonction de décision de manière plus concise : $h(x) = \text{sign}(w^T \cdot x + b)$ puisque $w^T \cdot x + b > 0$ pour tous les points situés au-dessus de l'hyperplan et $w^T \cdot x + b < 0$ pour tous les points au-dessous.

Pour un problème linéairement séparable, cf le diagramme ci-dessous, l'idée est de trouver l'hyperplan (appelé aussi Frontière de décision) qui maximise la marge. Ou encore, l'hyperplan tel que l'écart entre la frontière de décision et les points les plus proches (également appelé(s) vecteurs de support "support vectors") de l'une des classes soit maximal. Formellement, tout point x_i labelée y_i vérifie $h(x_i)y_i > 0$. Un tel séparateur est appelé **Maximal Marginal Hyperplane (MMH)**. Il s'agit donc de garder les classes le plus loin possible de la frontière de décision pour minimiser l'erreur de classification.

On considère l'hyperplan $w^T \cdot x + b = 1$ tel que tous les points au-dessus sont classé $+1$ et de même hyperplan $w^T \cdot x + b = -1$ tel que tous les points au-dessous sont classé -1 . Donc, on peut vérifier que les données ont été complètement séparables si elles vérifient tous la condition $h(x_i)y_i > 1$. Ces deux hyperplans sont parallèles, donc on peut

facilement calculer la distance entre eux : $\frac{|(b-a)-(b+1)|}{\|w\|} = \frac{2}{\|w\|}$. Maximiser cette distance revient à minimiser $\frac{\|w\|}{2}$ ou encore minimiser $\frac{w^T w}{2}$. Finalement on définit notre problème d'optimisation :

$$\min_{w,b} \frac{w^T w}{2}$$

tel que

$$h(x_i)y_i \geq 1 \forall i = 1, 2, \dots, m.$$

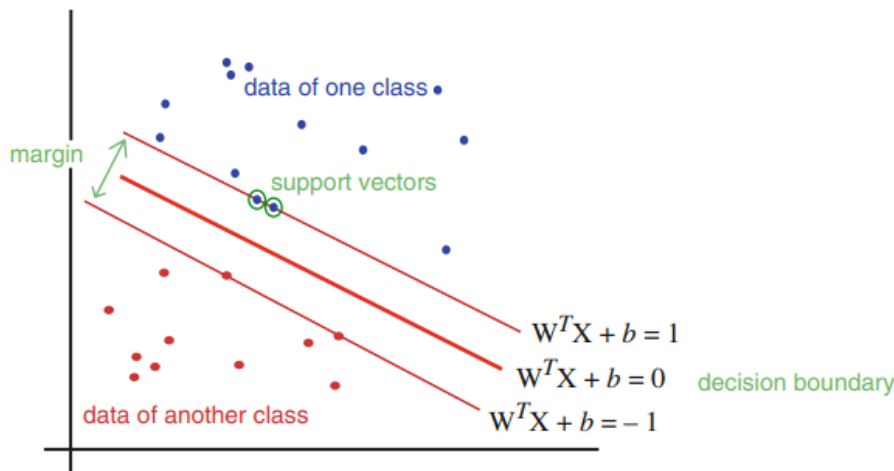


FIGURE 2 – Représentation des données dans un espace linéairement séparable à deux dimensions

Formulation de marge faible (Soft-Margin Extension) : Un hyperplan maximal n'est pas toujours l'hyperplan optimal permettant une classification rigoureuse surtout dans le cas de présence de points aberrants. La motivation pour avoir la formulation de marge faible est donc de : permettre au SVM de faire un certain nombre d'erreurs et de garder une marge aussi large que possible afin que d'autres points puissent encore être classés correctement. Ce peut se traduire simplement par une modification de la fonction objective :

$$\min_{w,b} \frac{w^T w}{2} + C \cdot (\text{erreur})$$

C est un hyperparamètre qui décide du compromis entre la maximisation de la marge et la minimisation des erreurs. Lorsque C est petit, les erreurs de classification ont moins d'importance et l'accent est mis sur la maximisation de la marge, alors que lorsque C est grand, l'accent est mis sur la prévention des erreurs de classification au détriment de la marge. À ce stade, nous devons toutefois noter que toutes les erreurs ne sont pas égales. Les points de données qui sont éloignés du mauvais côté de la frontière de décision doivent être davantage pénalisés que ceux qui sont plus proches. Voyons comment cela peut être intégré à l'aide du diagramme suivant.

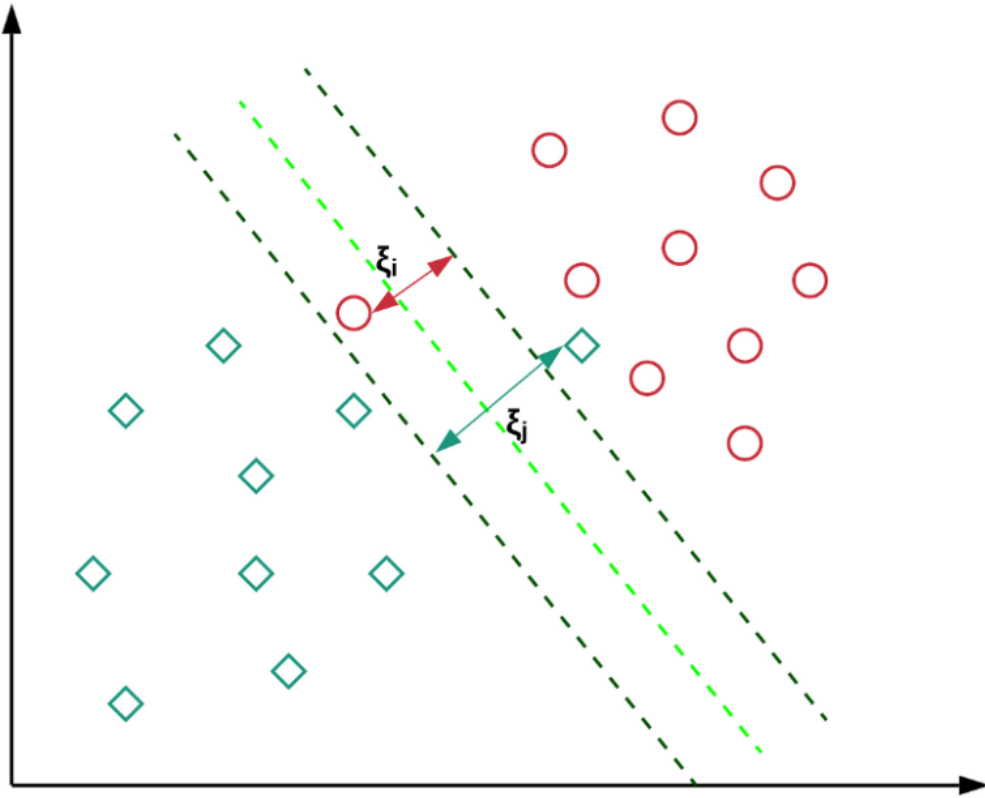


FIGURE 3 – La pénalité encourue par les points de données du fait qu'ils se trouvent du mauvais côté de la frontière de décision.

L'idée est la suivante : pour chaque point de données x_i , nous introduisons une variable ξ_i , "slack variable". La valeur de ξ_i est la distance de x_i de la marge de la classe correspondante si x_i est du mauvais côté de la marge, sinon zéro. Ainsi, les points qui sont plus éloignés de la marge du mauvais côté seraient plus pénalisés. Notre problème de pénalisation devient :

$$\min_{w,b} \frac{w^T w}{2} + C \sum_{k=1}^m \xi_i$$

tel que $h(x_i)y_i \geq 1 - \xi_i$, $\xi_i \geq 0$ pour tout $i = 1, 2, \dots, m$.

Frontière de décision non linéaire :

Les points de données x_i peuvent ne pas être linéairement séparables dans l'espace original. Cependant, si nous projetons les points de données dans un espace de plus grande dimension, ils peuvent être linéairement séparables dans ce nouvel espace. C'est ce qu'on appelle le "kernel-trick", ou "l'astuce du noyau". Pour effectuer cette transformation de dimension, on applique aux entrées x une fonction non-linéaire $\phi(x)$. Par conséquent, nous réécrivons le problème d'optimisation comme suit :

$$\min_{w,b} \frac{w^T w}{2} + C \sum_{k=1}^m \xi_i$$

tel que $h(\phi(x_i)).y_i \geq 1 - \xi_i$, $\xi_i \geq 0$ pour tout $i = 1, 2, \dots, m$.

- Exemple de fonction ϕ :

$$\phi : R^2 \rightarrow R^3$$

$$(x_1, x_2) \rightarrow (z_1, z_2, z_3) := (x_2, \sqrt{2}x_2x_1, x_3)$$

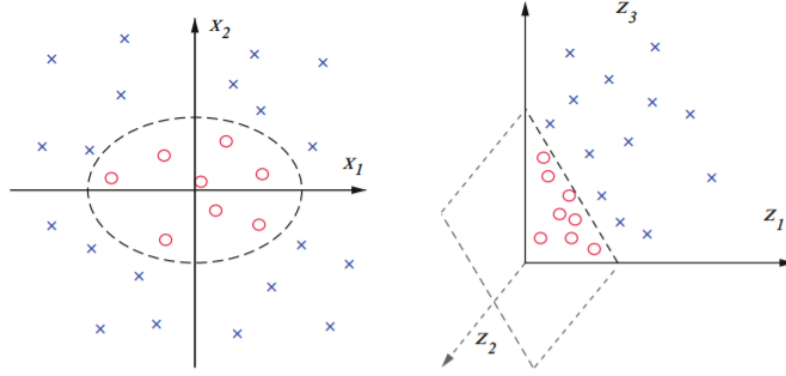


FIGURE 4 – Augmentation de la dimension

Formulation sous forme d'optimisation lagrangienne :

Nous réécrivons la fonction objectif comme :

$$\min_{w,b} \frac{w^T w}{2} + \sum_{k=1}^m \max_{\alpha_i \geq 0} \alpha_i (1 - y_i (w^T \cdot \phi(x_i) + b))$$

On a $\xi_i \geq 1 - h(\phi(x_i)) \cdot y_i$. Donc pour que ξ_i soit négligeable il faut que $1 - h(\phi(x_i)) \cdot y_i$ soit presque nulle. D'où l'ajout de la quantité $\max_{\alpha_i \geq 0} \alpha_i (1 - y_i (w^T \cdot \phi(x_i) + b))$. Car si $y_i (w^T \cdot \phi(x_i) + b) \geq 1$ α_i tend vers 0 sinon elle tend vers ∞ . Ce qui nous permet de pénaliser les points mal classés car $\max_{\alpha_i \geq 0} \alpha_i (1 - y_i (w^T \cdot \phi(x_i) + b))$ devient positive et importante.

Pour rester dans le cadre de la Formulation de marge faible, on ajoute la contrainte $\alpha_i \in [0, C]$. Ensuite, on jouant sur l'ordre du \max et \min dans la fonction objective on trouve :

$$\max_{\alpha_i \geq 0} [\min_{w,b} J(w, b, \alpha)]$$

avec

$$J(w, b, \alpha) = \frac{w^T w}{2} + \sum_{k=1}^m \alpha_i (1 - y_i (w^T \cdot \phi(x_i) + b))$$

Donc, maintenant il reste à minimiser J par rapport à w , b et ensuite maximiser le résultat par rapport à α .

$$\frac{\partial J}{\partial w} = 0 \implies w = \sum_{k=1}^m \alpha_i y_i \phi(x_i)$$

$$\frac{\partial J}{\partial b} = 0 \implies \sum_{k=1}^m \alpha_i y_i = 0$$

Finalement, on obtient :

$$\max_{\alpha_i \geq 0} \left[\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j) \right]$$

Avec $\sum_{k=1}^m \alpha_i y_i = 0$, $0 \leq \alpha_i \leq C$

Kernel Trick :

Cette optimisation pose un problème, elle fait intervenir des produits scalaires dans des espaces de grandes dimensions, ce qui est coûteux. La solution fait intervenir l'astuce du noyaux dont on parlait plus haut. Il faut utiliser une fonction noyaux (un opérateur intégral), vérifiant : $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$, on trouve ainsi

$$h(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + b$$

On se ramène ainsi à un problème dans l'espace d'origine, ce qui est beaucoup moins coûteux. On n'a également pas besoin de connaître explicitement la fonction ϕ , simplement la fonction noyau K .

Dans la pratique, les noyaux employés sont souvent le noyaux polynomial ou gaussien.

$$\text{Noyau gaussien : } K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$$

$$\text{Noyau polynomial : } K(x_i, x_j) = (x_i^T \cdot x_j + 1)^d$$

Donc, nous pouvons réécrire le problème dual comme :

$$\max_{\alpha_i \geq 0} \left[\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \right]$$

Avec

$$\sum_{k=1}^m \alpha_i y_i = 0, 0 \leq \alpha_i \leq C$$

En résolvant ce problème, on trouve α et b . Puis on peut classifier une donnée test x en calculant $h(x) = \text{sign}(\sum_{i=1}^m \alpha_i y_i K(x_i, x) + b)$.

3.1.2.3 Extreme Learning Machine (ELM)

L'approche ELM a été introduite pour surmonter certains problèmes de formation des réseaux de rétropropagation (backpropagation network), en particulier les taux de convergence potentiellement lents, l'ajustement critique des hyperparamètres d'optimisation, et la présence de minimums locaux qui nécessitent des stratégies de ré-entraînement multiples.

L'entraînement d'un modèle ELM nécessite un ensemble d'entraînement, X , de N paires labellisées, (x_i, y_i) , où $x_i \in R^m$ est le i -ième vecteur d'entrée et $y_i \in R$ est la valeur attendue associée. L'utilisation d'une sortie scalaire implique que le réseau a une seule unité de sortie (output). Les modèles ELM sont construits d'une seule couche cachée qu'on suppose contient N_h neurones connectées avec les m neurones de la couche d'entrée par des poids $w_j \in R^m, j = 1, \dots, N_h$. On associe à chaque neurone de la couche cachée un biais, paramètre très utile dans la théorie de réseaux de neurones. L'activation, représente en

général la réponse d'un neurone à un signal d'entrée, d'un neurone de la couche cachée est définie par :

$$a_j = \phi(\hat{w}_j \cdot x + \hat{b}_j)$$

Avec ϕ fonction d'activation et $x \in R^m$ un vecteur d'entrée (input vector).

La sortie du réseau est tous simplement une combinaison linéaire des sorties pondérées des neurones de la couche cachée :

$$f(x) = \sum_{j=1}^{N_h} \tilde{w}_j a_j(x)$$

En pratique on définit ce qu'on appelle la matrice d'activation H tel que ses éléments $h_{ij}, i = 1, \dots, m, j = 1, \dots, N_h$.

La particularité de ce modèle c'est que les quantités (\hat{w}_j, \hat{b}_j) sont aléatoirement choisis et ne nécessite aucune modification. Par contre, ce qu'il faut ajuster ce sont les quantités (\tilde{w}_j, \tilde{b}) . L'entraînement du modèle est en fait réduit en un problème de minimisation :

$$\min_{\tilde{w}, \tilde{b}} ||H\tilde{w} - y||$$

Une pseudo-inversion matricielle donne l'unique solution du problème :

$$\tilde{w} = H^+ y$$

Pour résumer, La procédure simple et efficace d'entraînement d'un ELM comprend les étapes suivantes 1. Fixer aléatoirement les poids d'entrée \hat{w}_i et le biais \hat{b}_i pour chaque neurone caché ; 2. Calculer la matrice d'activation, H , comme indiqué ; 3. Calculer les poids de sortie en résolvant un problème de pseudo-inverse selon.

1. Fixer aléatoirement les poids d'entrée \hat{w}_i et le biais \hat{b}_i pour chaque neurone de la couche cachée.
2. Calculer la matrice d'activation, H .
3. Calculer les poids de sortie en résolvant un problème de pseudo-inverse selon l'équation précédente.

Malgré l'apparente simplicité de l'approche ELM, l'expérience a montré que même le fait d'utiliser des poids aléatoires dans la couche cachée confère à un réseau avec une capacité de représentation notable. Dans le but de privilégier une solution particulière dotée de propriétés qui semblent pertinentes, on utilise la régularisation de Tikhonov, qui permet d'introduire un terme de test dans la minimisation :

$$\tilde{w} = H^+ y + ||\Gamma x||^2$$

3.2 Topic Modeling

Le Topic Modeling est une approche fondée sur des principes permettant de découvrir des thèmes (topics) à partir d'un grand corpus de documents sous forme textuel. Les résultats les plus courants d'un modèle Topic Model sont un ensemble de groupes de mots (clusters) et une distribution de thèmes pour chaque document. Chaque groupe de mots (cluster) est appelé un sujet et est une distribution de probabilité sur les mots (également appelés termes topiques) dans le corpus. La distribution des sujets d'un document donne

la proportion de chaque sujet dans le document. Il existe deux modèles de sujet de base l'**Analyse Sémantique Latente Probabiliste (pLSA)** (Hofmann, 1999) et **Latent Dirichlet Allocation** (Blei, 1999). Ce sont toutes deux des méthodes non supervisées. Bien qu'elles soient principalement utilisées pour modéliser et extraire des sujets de documents textuels, elles peuvent être étendues pour modéliser de nombreux autres types d'informations.

3.2.1 Latent Dirichlet Allocation

LDA est un modèle d'apprentissage non supervisé qui suppose que **"chaque document est constitué d'un mélange de sujets et que chaque sujet est une distribution de probabilité sur les mots."**

En général, il s'agit d'un modèle qui spécifie une procédure probabiliste par laquelle les documents sont générés.

La figure ci-dessous représente le modèle graphique de LDA (Plate Notation). On commence d'abord par expliciter les paramètres du modèle :

- M réfère au nombre de documents
- N Nombre de mots dans un document
- K Nombre de thèmes/sujets
- V Nombre de mots en total dans le vocabulaire
- α paramètre de Dirichlet qui distributions de sujets par document
- β paramètre de Dirichlet qui distributions de mots par sujets
- θ_d distribution de sujets par document
- ϕ_k distribution de mots par sujet

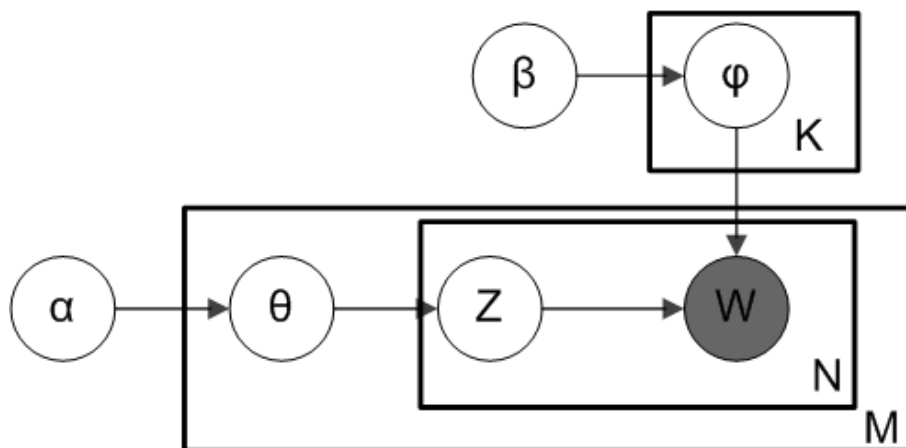


FIGURE 5 – Plate Notation qui représente le modèle LDA

Le modèle LDA est représenté comme un modèle graphique probabiliste dans la figure 1. Comme le montre la figure, la représentation LDA comporte trois niveaux. Les paramètres α et β sont des paramètres de niveau corpus, supposés être échantillonnés une fois dans le processus de génération d'un corpus. Les variables θ_d sont des variables au niveau du document, échantillonnées une fois par document. De même, les variables ϕ_k sont des variables au niveau du sujet, échantillonnées une fois par sujet. Enfin, les variables z_{ij} et w_{ij} sont des variables au niveau des mots, échantillonnées une fois pour chaque mot dans chaque document.

Corpus : $D = \{w_1, w_2, \dots, w_m\}$
 $w_i = (w'_{i1}, \dots, w'_{iN_1})$: i ème document du Corpus.
 w'_{ij} est le j ième mot du document i peut être représenté par un couple (z_{ij}, w_{ij})
 — z_{ij} est le sujet du mot
 — w_{ij} est le mot mais cette fois-ci représenter par un vecteur de taille V qui contient 0 partout sauf pour la position du mot dans le vocabulaire il contient le 1.

Pour détecter les sujets d'un corpus, nous imaginons un processus génératif par lequel les documents sont , afin que nous puissions les détecter, ou faire de l'ingénierie inverse. Nous imaginons le processus génératif comme suit. Les documents sont représentés comme des mélanges aléatoires de sujets latents, où chaque sujet est caractérisé par une distribution sur tous les mots. LDA suppose le processus génératif suivant pour un corpus D de M documents chacun de longueur N_i (tel que : $\sum_{d=1}^M N_d = N$) :

1. $\theta_i \sim Dir(\alpha)$ ou Dir est une distribution de Dirichlet avec un paramètre symétrique α qui est généralement peu abondant (sparse) $\alpha < 1$. (cf Figure 7)
2. $\phi_k \sim Dir(\beta)$
3. Pour toute positions de mots i, j avec $i \in \{1, \dots, M\}$ et $j \in \{1, \dots, N_i\}$:
 — $z_{ij} \sim Multinomial(\theta_i)$
 — $w_{ij} \sim Multinomial(\phi_{z_{ij}})$

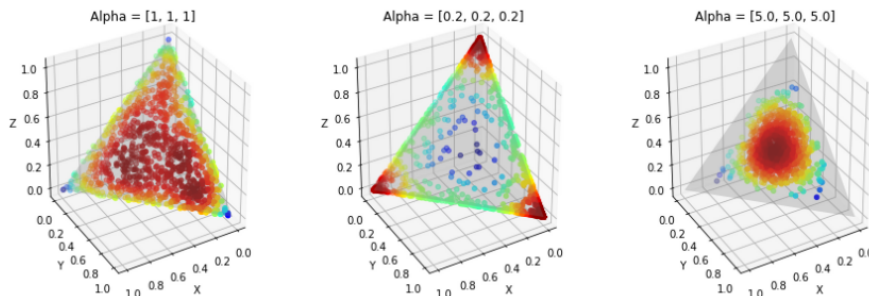


FIGURE 6 – Comment θ_i varie selon les valeurs de α

Loi de Dirichlet : Soit $\theta_d = (\dots, \theta_{dk}, \dots)$ une variable aléatoire suivant la loi de Dirichlet. θ_{dk} est la probabilité que le sujet k apparaîtra dans le document d , donc : $\theta_{dk} > 0$ et $\sum_{k=1}^K \theta_{dk} = 1$. La densité du θ_d est :

$$p(\theta_d | \alpha) = \frac{\Gamma(\sum_{i=1}^K \alpha_i)}{\prod_{i=1}^K \Gamma(\alpha_i)} \theta_{d1}^{\alpha_1} \dots \theta_{dK}^{\alpha_K}$$

Avec $\Gamma(x)$ la fonction Gamma. Cette distribution permet donc d'obtenir une distribution multinomiale de paramètre θ_d , correspondant pour LDA au mélange de topics d'un document d . La figure 7 montre la densité d'une telle distribution pour 3 sujets. Ici, chaque sommet du triangle réfère à un sujet, Lorsque $\alpha < 1$ les points se condensent dans les coins ce qui reviens à des θ_{dk} qui sont proches de 0 sauf quelques uns qui représente le sujet au quel le document appartient .

Probabilité d'un document (probability of a corpus) :

Etant donné les paramètres α et β , une distribution de topic par document et une distribution de mots par topic ϕ on a :

Probability of a document

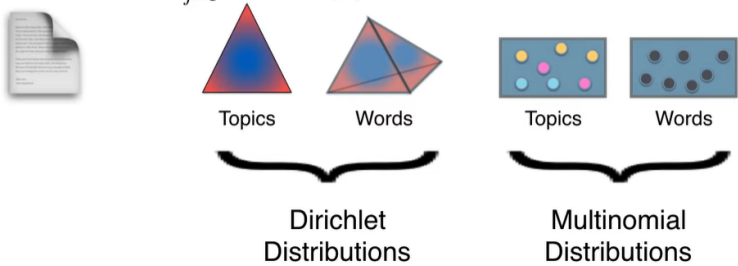
$$P(\mathbf{W}, \mathbf{Z}, \theta, \phi; \alpha, \beta) = \prod_{j=1}^M P(\theta_j; \alpha) \prod_{i=1}^K P(\phi_i; \beta) \prod_{t=1}^N P(Z_{j,t} | \theta_j) P(W_{j,t} | \phi_{Z_{j,t}})$$


FIGURE 7 – Probability of a corpus

Ce qui signifie quel est la probabilité de générer un des documents d sachant le corpus D et les paramètres α et β ?

Gibbs Sampling :

Dans le but d'identifier les distributions et ϕ , deux algorithmes ont été proposés : **Gibbs Sampling** et **Variational inference**.

Ces deux méthodes n'estime pas directement les distributions et ϕ mais ils estiment la distribution de Z : Affectation des sujet aux mots, tenant compte des mots w tout en marginalisant et ϕ . Tout z_i avec $i \in \{1, \dots, M\}$ renvoie à un topic auquel chaque mot w_i du document d est assigné. Et puis à partir de Z on peut calculer les distributions et ϕ .

L'idée clé derrière Gibbs Sampling est que pour obtenir un échantillon de la distribution conjointe des variables Z , nous échantillonnons chacune des variables aléatoires séparées z_i , conditionnellement à l'état de toutes les autres variables. La collection résultante de valeurs échantillonnées de z_i constitue un échantillon qui se rapproche de l'échantillon de la distribution conjointe de toutes les variables de Z .

Pour les modèles LDA basés sur le Gibbs Sampling, l'idée principale est de mettre à jour les topics pour chaque mot w du document d . Cela se traduit par la relation suivante :

$$p(z_i = j | Z_{-i}, W) \propto \frac{(n_{j,w}^{-i} + \beta_{j,w})(n_{d,j}^{-i} + \alpha_{d,j})}{\sum_{t=1}^T (n_{j,t}^{-i} + \beta_{j,t})}$$

Avec :

- Z_{-i} représente l'affectation des topics pour tout les mots du Corpus sauf pour le i ème.

- $n_{d,j}^{-i}$ représente nombre de fois le topic j à été assigné aux mots du document d à l'exclusion du mot i
- $n_{j,w}^{-i}$ représente nombre de fois le mot w à été assigné au topic j à l'exclusion du mot i .
- $\alpha_{d,j}$ le paramètre α varie selon le document et selon le topic, mais on peut considérer α constant.

```

Input: words  $\mathbf{w} \in$  documents  $\mathbf{d}$ 
Output: topic assignments  $\mathbf{z}$  and counts  $n_{d,k}, n_{k,w}$ , and  $n_k$ 
begin
  randomly initialize  $\mathbf{z}$  and increment counters
  foreach iteration do
    for  $i = 0 \rightarrow N - 1$  do
       $word \leftarrow w[i]$ 
       $topic \leftarrow z[i]$ 
       $n_{d,topic} = 1; n_{word,topic} = 1; n_{topic} = 1$ 
      for  $k = 0 \rightarrow K - 1$  do
         $p(z = k | \cdot) = (n_{d,k} + \alpha_k) \frac{n_{k,w} + \beta_w}{n_k + \beta \times W}$ 
      end
       $topic \leftarrow \text{sample from } p(z | \cdot)$ 
       $z[i] \leftarrow topic$ 
       $n_{d,topic} += 1; n_{word,topic} += 1; n_{topic} += 1$ 
    end
  end
  return  $\mathbf{z}, n_{d,k}, n_{k,w}, n_k$ 
end

```

FIGURE 8 – Algorithme de Gibbs Sampling pour un problème LDA

Après un grand nombre d'itérations de l'échantillonnage de Gibbs pour les mots de tous les documents, nous obtenons la distribution des topics par document θ_d et la distribution de mots par topic :

- Probabilité que topic t soit assigné au document d :

$$\hat{\theta}_{t,d} = \frac{(n_{t,d} + \alpha)}{\sum_{t'=1}^T (n_{t',d}^{-i} + \alpha)}$$

- Probabilité que le mot ν soit assigné au topic t :

$$\hat{\phi}_{\nu,t} = \frac{n_{\nu,t} + \beta}{\sum_{\nu'=1}^V (n_{\nu',t} + \beta)}$$

Dans la plupart des applications, l'utilisateur s'intéresse à $\phi_{\nu,t}$ qui donne une liste de mots (le vocabulaire) sous chaque sujet t , classés selon leurs probabilités dans $\phi_{\nu,t}$. Les mots les mieux classés donnent souvent une bonne indication de la classe du sujet. Par exemple, dans un ensemble de revues, LDA trouve les mots les mieux classés suivants pour un sujet : prix, vente, cher, coût, achat, marché. Nous pouvons voir que le sujet porte sur le prix du produit, et nous pouvons donc étiqueter le sujet avec prix.

4 Data Mining : approche espace vectoriel

L'exploration de textes ou le Data Mining est un nouveau domaine de l'informatique qui entretient des liens étroits avec le traitement du langage naturel (NLP), l'exploration de données, l'apprentissage automatique, la recherche d'informations et la gestion des connaissances. Le Data Mining vise à extraire informations utiles de données textuelles non structurées en identifiant et en explorant des patterns intéressants.

4.1 L'approche TF-IDF

TF-IDF signifie "Term Frequency - Inverse Document Frequency". Il s'agit d'une technique permettant de quantifier un mot dans les documents, nous calculons généralement un poids pour chaque mot qui signifie l'importance du mot dans le document et le corpus. Cette méthode est largement utilisée dans le domaine de la Information Retrieval et de Text Mining. Pour plus d'information référer à la sous-section 2.2.2.

4.2 Word Embedding : Word2vect

Word Embedding est l'une des représentations les plus populaires du vocabulaire des documents. Elle est capable de capturer le contexte d'un mot dans un document, la similarité sémantique et syntaxique, la relation avec d'autres mots, etc. Et cela en associant à chaque mot du corpus une représentation vectorielle.

Une des techniques les plus populaires pour effectuer le Word Embeddings est la technique **Word2Vec**. Elle a été développée par Tomas Mikolov en 2013 chez Google à l'aide d'un réseau neuronal peu profond (contenant une seule couche cachée).

Le modèle word2vec et ses applications ont attiré beaucoup d'attention ces deux dernières années. Les représentations vectorielles des mots apprises par les modèles word2vec se sont avérées porteuses de sens sémantiques et sont utiles dans diverses tâches NLP.

Word2vec peut utiliser l'une ou l'autre des deux architectures suivante pour produire une représentation distribuée des mots : **continuous bag-of-words CBOW** ou le **Continuous skip-gram**.

4.2.1 Continuous bag-of-words (CBOW) :

4.2.1.1 One-word context

Nous partons de la version la plus simple du modèle de sac de mots continu (CBOW). Nous supposons qu'il n'y a qu'un seul mot considéré par contexte, ce qui signifie que le modèle prédit un mot cible à partir d'un mot de contexte.

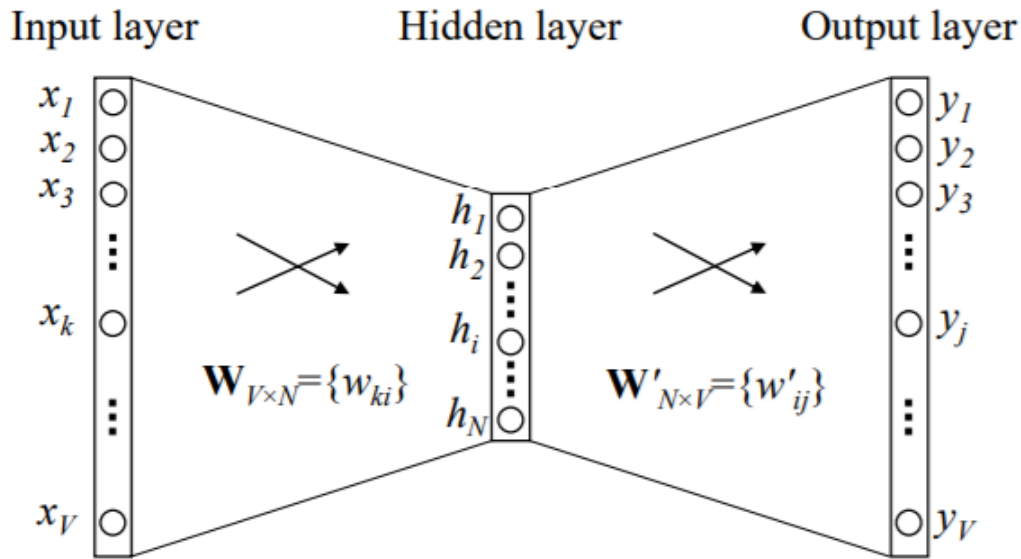


FIGURE 9 – Un modèle CBOW simple avec un seul mot dans le contexte

La figure 9 renvoie à ce modèle simplifié. On note V la taille du vocabulaire, et N la taille de la couche cachée. Les neurones de la couches cachées sont entièrement connectées avec les neurones de la couche d'entrée et de sortie. L'entrée est un vecteur **one-hot**, c'est-à-dire que pour un mot de contexte en entrée, une seule des V valeurs, (x_1, \dots, x_V) , vaut 1, et toutes les autres valeurs valent 0. $W_{V \times N}$ est la matrice des poids interne entre la couche d'entrée et la couche cachée et $W'_{N \times V}$ la matrice des poids interne entre la couche cachée et la couche de sortie. Formellement, en note la ligne i de la matrice $W_{V \times N}$ v_w^T et la colonne i de la matrice $W'_{N \times V}$ v'_w .

$v_w \in R^N$ et $v'_w \in R^N$ sont deux représentations différentes

du mot de contexte w . Le premier est appelé **“input vector”**

et le deuxième **“output vector”** du mot de contexte w .

Etant donné un mot de contexte $w = (x_1, \dots, x_V)$ tel que $x_k = 1$ et $x'_k = 0$ pour $k \neq k'$, on a :

$$h = W^T \cdot x = W_{(:,k)}^T = v_{w_I}^T \quad (1)$$

$v_{w_I}^T$ étant le vecteur représentant le mot de contexte en entrée w_I . L'équation 1 signifie que la fonction d'activation choisi entre l'Input Layer et le Hidden Layer est simplement linéaire. Ce n'est pas le cas pour la fonction d'activation considérée entre le Hidden Layer et le Output Layer qui est la fonction **Softmax**. De même, e

En utilisant la matrice des poids $W'_{N \times V}$ on calcule le score u_j de chaque mot appartenant au vocabulaire :

$$u_j = v_{w_j}'^T \cdot h \quad (2)$$

Puis on applique la fonction d'activation sur le score pour obtenir la distribution des mots :

$$p(w_j | w_I) = y_j = \frac{\exp(u_j)}{\sum_{j'=1}^V \exp(u_{j'})} \quad (3)$$

Avec y_j est la j ème composante du vecteur de sortie y et représente $p(w_j | w_I)$ la probabilité que le mot w_j soit le mot cible recherché.

En remplaçant 1 et 2 dans 20, on obtient :

$$p(w_j|w_I) = y_j = \frac{\exp(v'_{w_j} \cdot v_{w_I})}{\sum_{j'=1}^V \exp(v'_{w_{j'}} \cdot v_{w_I})} \quad (4)$$

- Équation de mise à jour des poids cachés-sorties

En utilisant l'algorithme de descente du gradient stochastique, nous obtenons l'équation de mise à jour des poids w'_{ij}

$$w'_{ij}^{(new)} = w'_{ij}^{(old)} - \eta \cdot e_j \cdot h_i \quad (5)$$

ou encore :

$$v'_{w_j}^{(new)} = v'_{w_j}^{(old)} - \eta \cdot e_j \cdot h \quad \forall j = 1, 2, \dots, V \quad (6)$$

- $\eta > 0$: le taux d'apprentissage.
- e_j : l'erreur de prédiction de la couche de sortie. Il est calculé en dérivant la fonction de perte (loss function) E par rapport au j ième score u_j :

$$\frac{\partial E}{\partial u_j} = y_j - t_j := e_j \quad (7)$$

Avec :

$$E = -\log(p(w_j^*|w_I)) \quad (8)$$

- $p(w_j^*|w_I)$ étant la probabilité conditionnelle d'observer le mot de sortie réel w_j^* étant donné le mot de contexte d'entrée w_i en tenant compte des poids.
- t_j ne sera égal à 1 que si w_j le mot de sortie prédit par le modèle est le mot de sortie réel, sinon $t_j = 0$.

Notez que l'équation 6 implique que nous devons passer en revue chaque mot possible du vocabulaire, vérifier sa probabilité de sortie y_j , et comparer y_j avec sa sortie attendue t_j (soit 0 ou 1).

- Si $y_j > t_j$ ("surestimation"), alors nous soustrayons une proportion du vecteur caché h (ie v_{w_I}) de v'_{w_j} rendant ainsi v'_{w_j} plus éloigné de v'_{w_I} dans l'espace des représentations vectorielles des mots.
- si $y_j < t_j$ (" sous-estimation ", ce qui n'est vrai que si $t_j = 1$, c'est-à-dire, $w_j = w_{j^*}$), on ajoute une fraction de h à v'_{w_j} rendant ainsi v'_{w_j} plus proche de v_{w_I} .
- Si y_j est très proche de t_j , alors, selon l'équation de mise à jour, les poids seront très peu modifiés.

Notez, encore une fois, que v_w (input vector) et v'_w (output vector) sont deux représentations vectorielles différentes du mot w .

- Équation de mise à jour des poids entrées-cachés

Ayant obtenu les équations de mise à jour pour W' nous pouvons maintenant passer à W . Nous prenons la dérivée de E sur la sortie de la couche cachée, on écrit :

$$\frac{\partial E}{\partial h_i} = \sum_{j=1}^V \frac{\partial E}{\partial u_j} \frac{\partial u_j}{\partial h_i} = \sum_{j=1}^V e_j \cdot w'_{ij} := EH_i \quad (9)$$

EH , un vecteur de taille N , est la somme des vecteurs de sortie de tous les mots du vocabulaire, pondérés par leur erreur de prédiction :

$$EH = \sum_{j=1}^V e_j v'_j \quad (10)$$

Or d'après 1 on a :

$$h_i = \sum_{j=1}^V x_k \cdot w_{ki}$$

Donc la dérivée de E par rapport au éléments de W s'écrit :

$$\frac{\partial E}{\partial w_{ki}} = \frac{\partial E}{\partial h_i} \frac{\partial h_i}{\partial w_{ki}} = EH_i \cdot x_k \quad (11)$$

En utilisant l'algorithme de descente du gradient stochastique, nous obtenons l'équation de mise à jour des poids w_{ki} .

$$w_{ki}^{(new)} = w_{ki}^{(old)} - \eta \cdot EH_i \cdot x_k \quad (12)$$

Puisqu'une seule composante de x est non nulle $x_k = x_{w_I} = 1$:

$$w_{ki}^{(new)} = w_{ki}^{(old)} - \eta \cdot EH_k \quad (13)$$

Finalement, l'équation de mise à jour des poids w_{ij}

$$v_{w_I}^{(new)} = v_{w_I}^{(old)} - \eta \cdot EH \quad (14)$$

où v_{w_I} est une ligne de W , le "vecteur d'entrée" du seul mot de contexte, et représente aussi la seule ligne de W dont la dérivée est non nulle. Toutes les autres lignes de W restent inchangées après cette itération, car leurs dérivées sont nulles. cette itération, car leurs dérivées sont nulles.

Intuitivement, puisque le vecteur EH est la somme des vecteurs de sortie de tous les mots du vocabulaire pondérés par leurs erreurs de prédictions $e_j = y_j - t_j$, nous pouvons comprendre 14 comme l'ajout d'une partie de chaque vecteur de sortie des mots du vocabulaire v'_j au vecteur d'entrée du mot de contexte v_{w_I} . Si, dans la couche de sortie, la probabilité qu'un mot w_j soit le mot de sortie réel w_{j*} est surestimée ($y_j > t_j$), alors le vecteur d'entrée du mot de contexte v_{w_I} aura tendance à s'éloigner du vecteur de sortie de v_{w_j} . Inversement, si la probabilité que w_j soit le mot de sortie est sous-estimée ($y_j < t_j$), alors le vecteur d'entrée v_{w_I} aura tendance à se rapprocher du vecteur de sortie de v_{w_j} ; Si la probabilité de w_j est prédite de manière assez précise, alors elle aura peu d'effet sur le mouvement du vecteur d'entrée du mot de contexte v_{w_I} . Le mouvement du vecteur d'entrée de w_I est déterminé par l'erreur de prédiction de tous les vecteurs du vocabulaire; plus l'erreur de prédiction est grande, plus les effets d'un mot sur le mouvement du vecteur d'entrée du mot de contexte sont importants.

4.2.1.2 Multi-word context

Dans le modèle Multi-word context (cf 11), lors du calcul de la sortie de la couche cachée, au lieu de copier directement le vecteur d'entrée du mot de contexte, le modèle CBOW prend la moyenne des vecteurs d'entrées des mots de contexte, ainsi h s'écrit :

$$h = \frac{1}{C} W^T (x_1 + x_2 + \dots + x_C) = \frac{1}{C} (v_{w_1} + v_{w_2} + \dots + v_{w_C}) \quad (15)$$

Où C est le nombre de mots dans le contexte, $w_1 + w_2 + \dots + w_C$ sont les mots du contexte, et v_w est le vecteur d'entrée d'un mot w .

La fonction de perte est défini de façon similaire à 24 :

$$E = -\log(p(w_j^* | w_{I,1}, w_{I,2}, \dots, w_{I,C})) \quad (16)$$

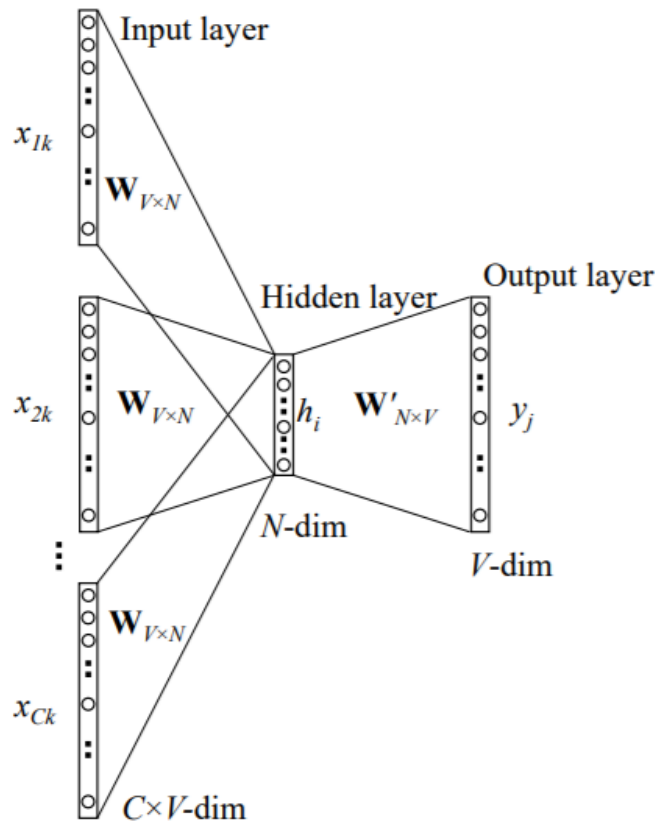


FIGURE 10 – Modèle CBOW pour plusieurs mots de contexte

L'équation de mise à jour pour les poids cachés-sortie reste la même que celle pour le modèle à seul mot de contexte :

$$v'_{w_j}{}^{(new)} = v'_{w_j}{}^{(old)} - \eta \cdot e_j \cdot h \quad \forall j = 1, 2, \dots, V \quad (17)$$

Cette équation est appliquée à chaque élément de la matrice W' pour chaque training instance.

L'équation de mise à jour des poids entrée→caché est similaire à l'équation 15, sauf que maintenant nous devons l'appliquer pour chaque mot $w_{I,C}$ dans le contexte :

$$v_{w_{I,c}}{}^{(new)} = v_{w_{I,c}}{}^{(old)} - \eta \cdot E H \quad \forall c = 1, 2, \dots, C \quad (18)$$

Où $v_{w_{I,c}}$ est le vecteur d'entrée du c-ième mot dans le contexte ; η est un taux d'apprentissage positif et $\frac{\partial E}{\partial h_i} = E H_i$.

4.2.2 Skip-Gram Model :

Le modèle Skip-Gram Model est l'inverse du modèle CBOW. Le mot cible est maintenant sur la couche d'entrée et les mots du contexte sont sur la couche de sortie. cf figure 11

Nous utilisons toujours v_{w_I} pour désigner le vecteur d'entrée du seul mot sur la couche d'entrée, nous avons donc la même définition des sorties de la couche cachée h :

$$h = W_{(.,k)}^T = v_{w_I}^T \quad (19)$$

Sur la couche de sortie, au lieu de générer une distribution multinomiale, nous sortons C distributions multinomiales. Chaque sortie est calculée en utilisant la même matrice cachée-sortie :

$$p(w_{c,j} = w_{c,j*} | w_I) = y_{c,j} = \frac{\exp(u_{c,j})}{\sum_{j'=1}^V \exp(u_{c,j'})} \quad (20)$$

Où $w_{c,j}$ est le j -ième mot sur le c -ième panneau de la couche de sortie ; $w_{c,j*}$ est le c -ième mot réel dans les mots du contexte de sortie ; w_I est le seul mot en entrée ; $y_{c,j}$ est la sortie de la j -ième unité sur le c -ième panneau de la couche de sortie ; $u_{c,j}$ est le score de la j -ième unité sur le c -ième panneau de la couche de sortie.

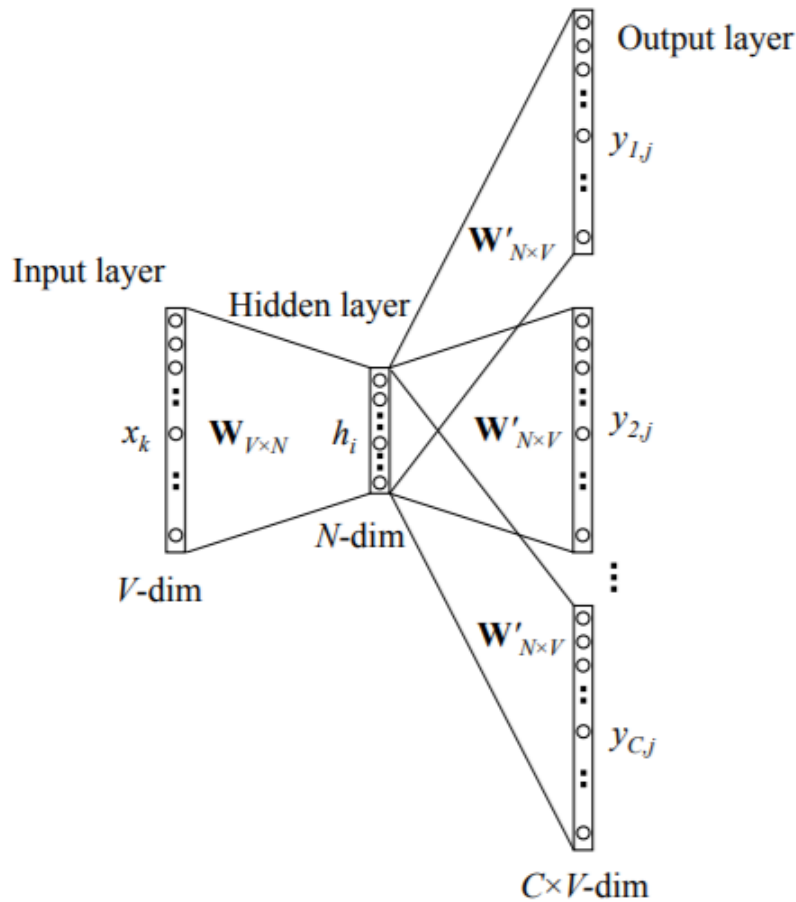


FIGURE 11 – Le modèle skip-gram

Le modèle suppose que tout les panneaux de sortie ont les mêmes poids :

$$u_{c,j} = u_j = v_{w_j}^{'T} \cdot h \quad \forall c = 1, 2, \dots, C \quad (21)$$

La fonction de perte est :

$$E = -\log(p(w_{j^*_1}, w_{j^*_2}, \dots, w_{j^*_C}) | w_I) \quad (22)$$

Avec j^*_c est l'indice du c -ième mot de contexte réel. L'erreur de prédiction est défini comme dans l'équation 7 avec une petite adaptation. On définit par ailleurs un vecteur de taille N EI qui représente la somme des erreurs de prédiction dans tout les contextes (ie. Les panneaux) tel que :

$$EI_j = \sum_{c=1}^C e_{c,j} \quad (23)$$

La dérivée de la matrice W' s'écrit :

$$\frac{\partial E}{\partial w'_{ij}} = \sum_{c=1}^C \frac{\partial E}{\partial u_{c,j}} \frac{\partial u_{c,j}}{\partial w'_{ij}} = \sum_{c=1}^C e_{c,j} \frac{\partial u_{c,j}}{\partial w'_{ij}} = EI_j \cdot h_i \quad (24)$$

Donc, l'équation de mise à jour des poids cachés-sortis :

$$w'_{ij}^{(new)} = w'_{ij}^{(old)} - \eta \cdot EI_j \cdot h_i \quad (25)$$

ou encore :

$$v_{w_j}^{'(new)} = v_{w_j}^{'(old)} - \eta \cdot EI_j \cdot h \quad \forall j = 1, 2, \dots, V \quad (26)$$

Notez que nous devons appliquer cette équation de mise à jour pour chaque élément de la matrice cachée-sortie pour chaque training instance.

La dérivation de l'équation de mise à jour pour la matrice entrée-caché est identique à 14 et 15, sauf qu'ici l'erreur de prédiction e_j est remplacée par EI_j . Nous donnons directement l'équation de mise à jour :

$$v_{w_I}^{(new)} = v_{w_I}^{(old)} - \eta \cdot EH \quad (27)$$

Avec EH un vecteur de N composantes chacune est défini comme suit :

$$EH_i = \sum_{j=1}^V EI_j \cdot w'_{ij} \quad (28)$$

5 Application : Analyse des Sentiments sur une base de données des Tweets

5.1 Introduction :

Dans cette partie, nous appliquons les approches de pré-traitement de textes vu dans la section 2.1, page 3 et la vectorization TF-IDF (cf paragraphe 2.2.2) à un corpus texte de polarité à trois sens (positif, négatif, neutre) pour les tweets, sans utiliser le moteur d'analyse de sentiment intégré à NLTK. Nous cherchons à produire une représentation numérique du corpus pour pouvoir appliquer les méthodes de Machine Learning : Random Forest Classifier, Naive Bayes, Support Vector Machines, Deep Learning.

5.2 Base de Données utilisées

- training.json : Ce fichier contient 15k tweets bruts, avec leurs étiquettes de polarité (1 = positif, 0 = neutre, -1 = négatif). Nous utiliserons ce fichier pour former nos classificateurs.
- develop.json : Dans le même format que training.json, le fichier contient un plus petit ensemble de tweets. Nous l'utiliserons pour tester les prédictions de nos classificateurs qui ont été entraînés sur le plateau d'entraînement.

| | Tweets | Preprocessed Tweets | labels |
|-------|---|---|--------|
| 0 | dear @Microsoft the newOoffice for Mac is grea... | [dear, microsoft, newooffice, mac, great, lync... | -1 |
| 1 | @Microsoft how about you make a system that do... | [microsoft, make, system, n't, eat, friggin, d... | -1 |
| 2 | If I make a game as a #windows10 Universal App... | [make, game, windows10, universal, app, xboxon... | 0 |
| 3 | Microsoft, I may not prefer your gaming branch... | [microsoft, prefer, game, branch, business, ma... | 1 |
| 4 | @MikeWolf1980 @Microsoft I will be downgrading... | [mikewolf1980, microsoft, downgrade, windows10... | -1 |
| ... | ... | ... | ... |
| 16800 | #WEB YouTube improves upload process with opti... | [web, youtube, improve, upload, process, optio... | 0 |
| 16801 | Gonna change my Tumblr theme. I hope I can fin... | [gon, na, change, tumblr, theme, hope, finish,... | 1 |
| 16802 | I'm so jealous of everyone at the Justin Biebe... | ['m, jealous, justin, bieber, concert, worry, ... | 0 |
| 16803 | Jim Harbaugh, Alex Smith Drive Giants World Se... | [jim, harbaugh, alex, smith, drive, giant, wor... | 0 |
| 16804 | #Trending: Tim Tebow is now dating cave woman ... | [trending, tim, tebow, date, cave, woman, 10,0... | 0 |

16805 rows × 3 columns

FIGURE 12 – DataFrame illustrant les données textuelles préprocessés

5.3 Échantillonnage des données non équilibrées

Les données déséquilibrées font généralement référence à un problème de classification où les classes ne sont pas représentées de manière égale. C'est le cas pour notre base de données où la classe 'Positive' représentée par le label +1 est majoritaire. Afin d'assurer le bon apprentissage de nos modèles, nous avons opté pour un échantillonnage des données comme le montre la figure 14 ci-dessous.

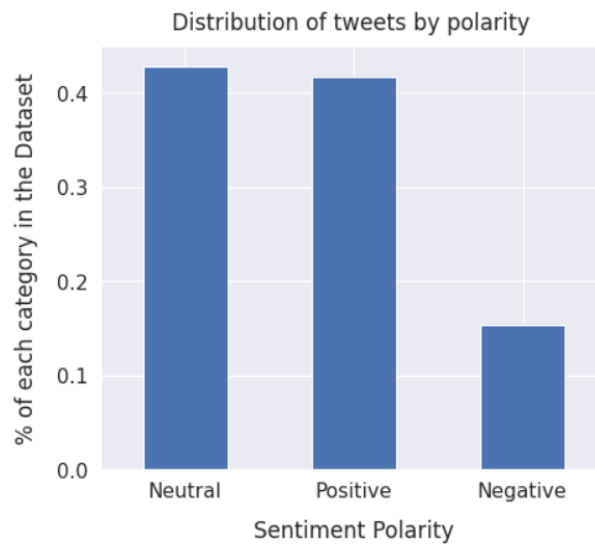


FIGURE 13 – Data déséquilibrée

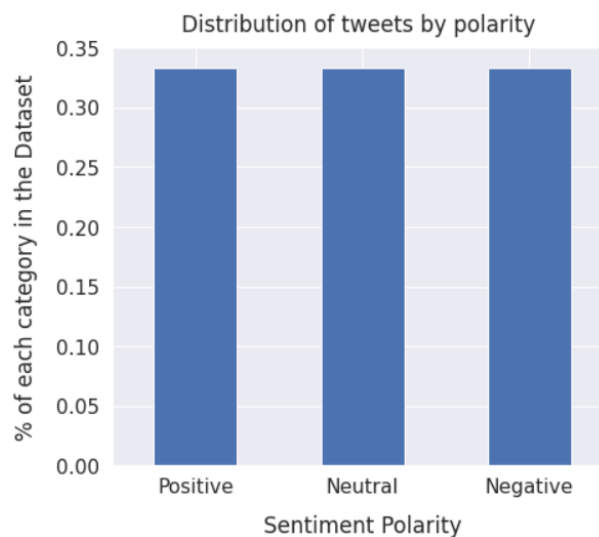


FIGURE 14 – Data équilibrée

5.4 Représentation numérique du Dataset par la méthode TF-IDF

Après avoir nettoyé la Dataset et la transformer en listes de "tokens" pour chaque tweet comme le montre la figure 12, on procède à une transformation **TF-IDF** (cf 2.2.2). On fait appel à la fonction prédéfini en **sklearn** : `TfidfVectorizer`. Les vecteurs résultants "Train_vectors" et "Test_vectors" sont :

```
array([[0., 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.,
        ...,
        [0.47367069, 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.,
        0., 0., 0., ..., 0., 0., 0.]])
```

FIGURE 15 – Train_vectors : représentation numérique du cleaned Train Data

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

FIGURE 16 – Test_vectors : représentation numérique du cleaned Test Data

Comme nous avons vu dans 2.2.2, La TF-IDF technique calcule la fréquence du terme dans chaque document en tenant compte de sa fréquence dans le corpus. C'est la raison pour laquelle les valeurs des éléments des vecteurs sont très inférieures à 0. Pour vérifier que le Train_vectors n'est pas un vecteur nul, on peut utiliser la méthode `.count_nonzero()` ou encore visualiser par la matrice des caractéristiques.

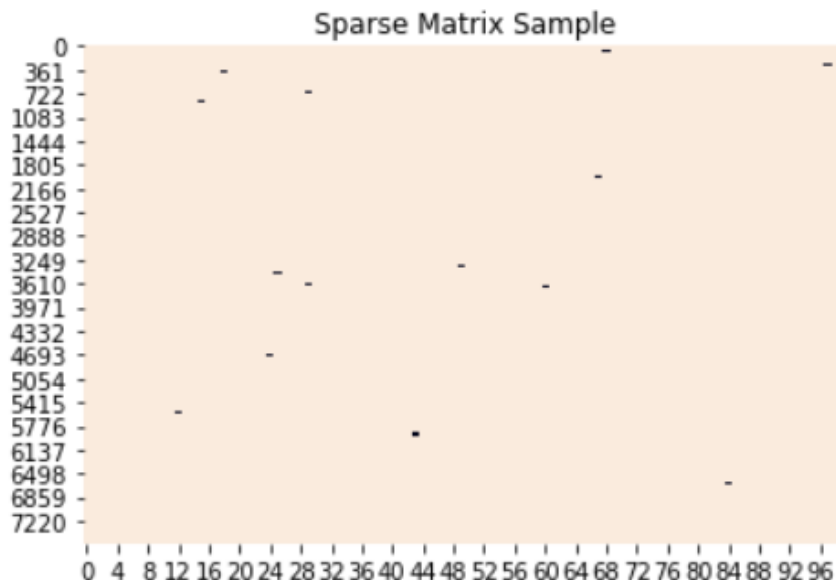


FIGURE 17 – Échantillon aléatoire de la matrice des caractéristiques (valeurs non nulles en noir)

5.5 Réduction de la dimensionnalité :

On utilise une technique linéaire de réduction de la dimensionnalité : **la décomposition en valeurs singulières (SVD)** qui projette linéairement les données dans un espace de dimension inférieure de telle sorte que la variance des données dans la représentation de dimension inférieure soit maximisée.

Une telle technique est déjà implémentée dans la librairie Sklearn : **TruncatedSVD**. On présente dans le tableau ci-dessous comment varie la variance des données selon le nombre de features considérées (`N_components`).

| <code>N_components</code> | Percentage of data | <code>explained_variance_ratio_.sum()</code> |
|---------------------------|--------------------|--|
| 10325 | 60% | 100% |
| 7560 | 43% | 100% |
| 6883 | 40% | 99,4% |
| 5162 | 30% | 99,4% |
| 3442 | 20% | 84,26% |

FIGURE 18 – Tableau variance expliqué en fonction de la dimension du problème

Dans la suite, on réduit la taille de la Data à **7560 features**.

5.6 Résultats

Le rapport de classification représente les mesures clés d'un problème de classification : precision, recall, f1-score. La précision et le recall sont en fait très utilisés pour les bases de données déséquilibrés, car dans une base de données très déséquilibré, une précision de 99 % peut être dénuée de sens.

- **Precision** : La précision est la capacité d'un classificateur à ne pas étiqueter comme positive une instance qui est en réalité négative. Pour chaque classe, elle est définie comme le rapport entre les vrais positifs et la somme des vrais positifs et des faux positifs.
- **Recall** : Le Recall est la capacité d'un classificateur à trouver toutes les instances positives. Pour chaque classe, il est défini comme le rapport entre les vrais positifs et la somme des vrais positifs et des faux négatifs.
- **F1-score** : la moyenne harmonique entre la précision et le Recall.
- **Support** : Le nombre d'occurrences réelles de la classe dans le jeu de données spécifié. Le support ne change pas d'un modèle à l'autre mais diagnostique plutôt le processus d'évaluation.

5.6.1 Méthode basée sur l'approche TF-IDF

D'après les matrices de confusions et les rapports de classification 19, 20, et 21 on remarque que le modèle SVC et Random Forest Classifier sont plus performants à prédire la classe "1" que les deux autres classes. Par exemple, pour le modèle " Support Vector Classifier " 70,98% des classes prédites comme "1" sont réellement labellisé comme "1". Toutefois, le modèle Réseau de Neurones Profonds prédit bien la classe "0" avec un score 55% alors qu'il fonctionne mal avec les deux autres classes. En réalité, le modèle a labellisé toute les tweets comme "0" c'est pour cela le f1-score des deux classes "-1" et "1" est nulle.

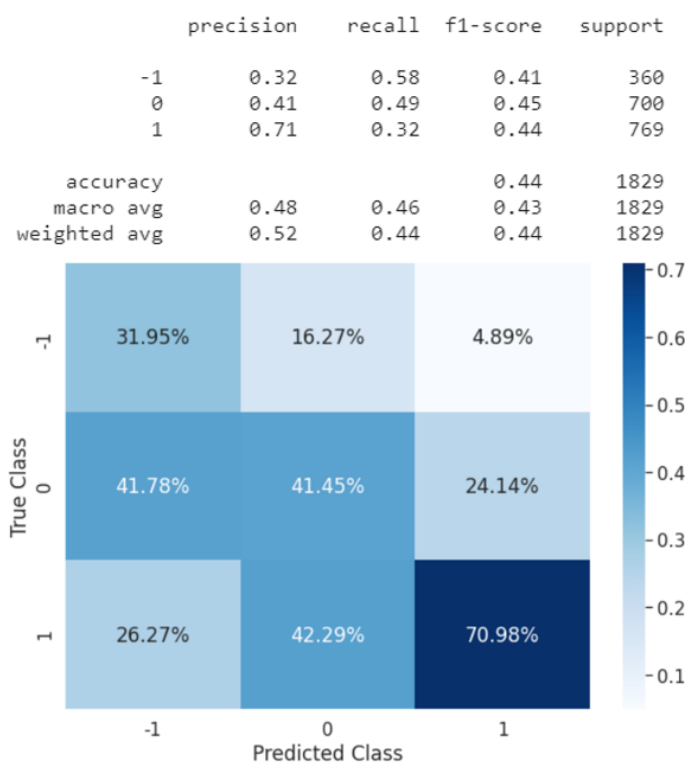


FIGURE 19 – TF-IDF approach : Classification Report and Confusion Matrix for the SVC model

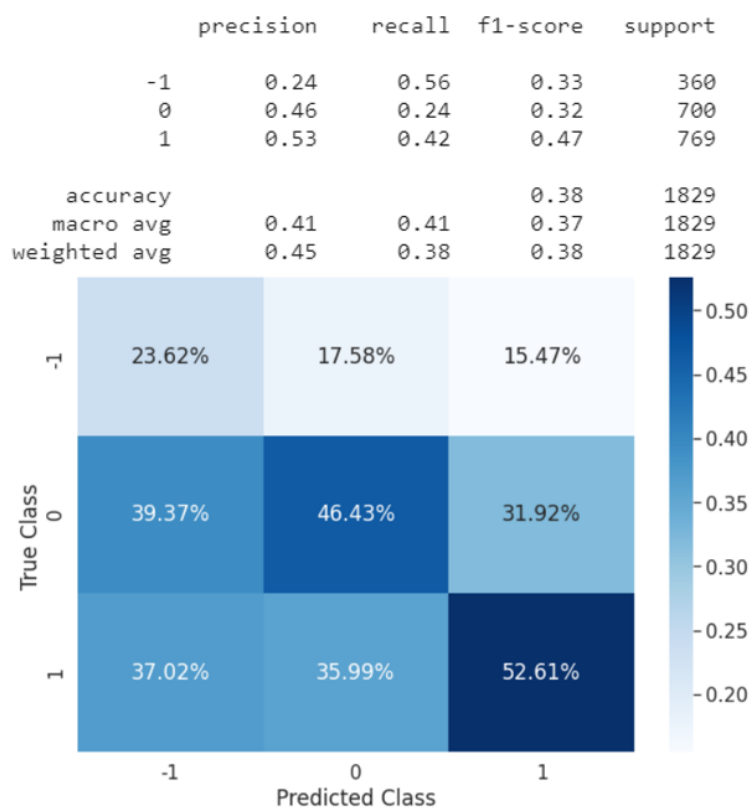


FIGURE 20 – TF-IDF approach : Classification Report and Confusion Matrix for the Random Forest model

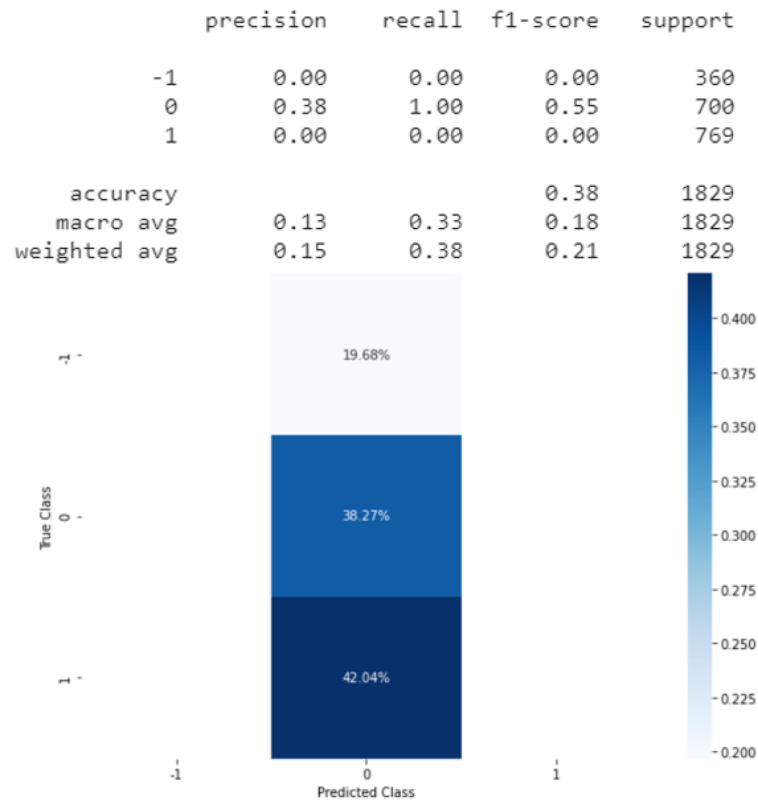


FIGURE 21 – TF-IDF approach : Classification Report and Confusion Matrix for the Deep Learning model

5.6.2 Méthode basée sur l'approche word2vec

En utilisant la méthode word2vec, vous n'avez pas besoin de procéder à une réduction de dimensionnalité; vous pouvez choisir le nombre de features à considérer à l'aide du paramètre `vector_size` du module **word2vec** prédéfini dans la librairie **gensim** qui représente le nombre de neurones dans la couche cachée N (Voir la figure 9). Nous avons choisi : $vector_size = 700$

Les résultats fournis 22 et 23 par l'approche word2vec sont proches à ceux fournis par l'approche TF-IDF.

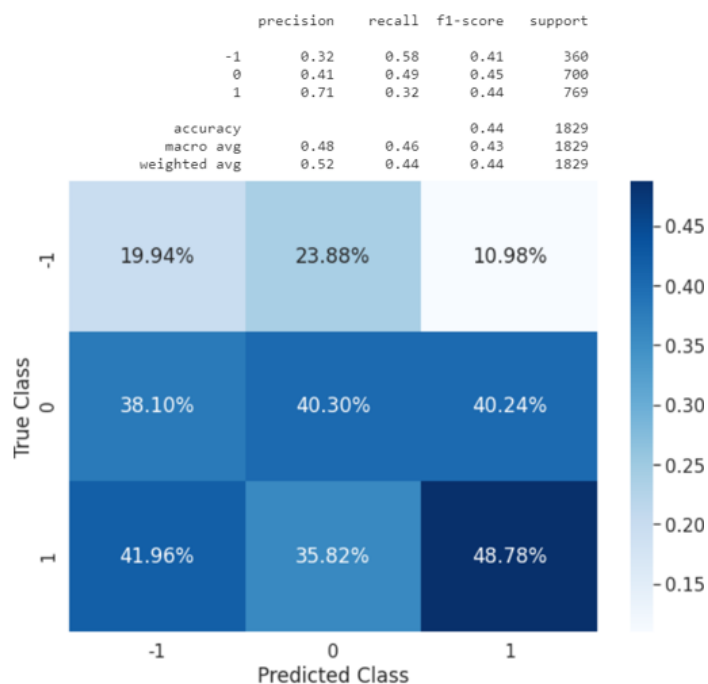


FIGURE 22 – Word2vec approach : Classification Report and Confusion Matrix for Support Vector Classifier

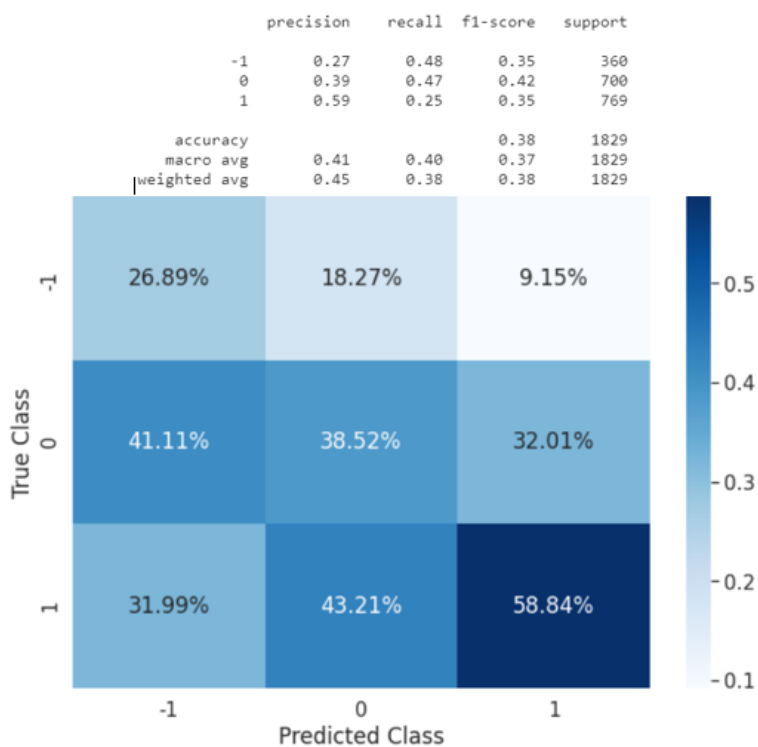


FIGURE 23 – Word2vec approach : Classification Report and Confusion Matrix for Random Forest Classifier

6 Conclusion

Malgré tous les défis et les problèmes potentiels qui menacent l'analyse des sentiments, on ne peut ignorer la valeur qu'elle ajoute à l'industrie. Parce que l'analyse des sentiments fonde ses résultats sur des facteurs intrinsèquement humains, elle est appelée à devenir l'un des principaux moteurs de nombreuses décisions commerciales à l'avenir.

7 Remerciements

Je tiens à saisir cette occasion afin d'adresser mes sincères remerciements à mon encadrant : M. Alexandre Saidi, pour sa disponibilité, pour son encadrement très enrichissant et ses conseils constructifs, qui ont contribué à la réussite de mon projet de recherche et à atteindre mes objectifs.

8 Bibliographie

<https://1lib.fr/book/3661774/c30be4?id=3661774&secret=c30be4>
<https://arxiv.org/pdf/1411.2738v3.pdf>
https://akuz.me/pdfs/akuz_lda_asym.pdf
<http://alberto.bietti.me/files/rapport-lda.pdf>
<https://u.cs.biu.ac.il/~89-680/darling-lda.pdf>
https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
<https://www.youtube.com/watch?v=T05t-SqKArY>
https://www.youtube.com/watch?v=BaM1uiCpj_E&t=1s
<https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-late>
<https://radimrehurek.com/gensim/models/word2vec.html> [https://books.google.fr/books?hl=fr&lr=&id=vOUBEAAAQBAJ&oi=fnd&pg=PR11&dq=Mining+Opinions,+Sentiments,+and+Emotions&ots=UfF-E_bFyN&sig=yYcnJLtyvJpFiCb5EbYyecfWr7c#v=onepage&q=Mining%\\$20Opinions%\\$2C%\\$20Sentiments%\\$2C%\\$20and%\\$20Emotions&f=false](https://books.google.fr/books?hl=fr&lr=&id=vOUBEAAAQBAJ&oi=fnd&pg=PR11&dq=Mining+Opinions,+Sentiments,+and+Emotions&ots=UfF-E_bFyN&sig=yYcnJLtyvJpFiCb5EbYyecfWr7c#v=onepage&q=Mining%$20Opinions%$2C%$20Sentiments%$2C%$20and%$20Emotions&f=false)