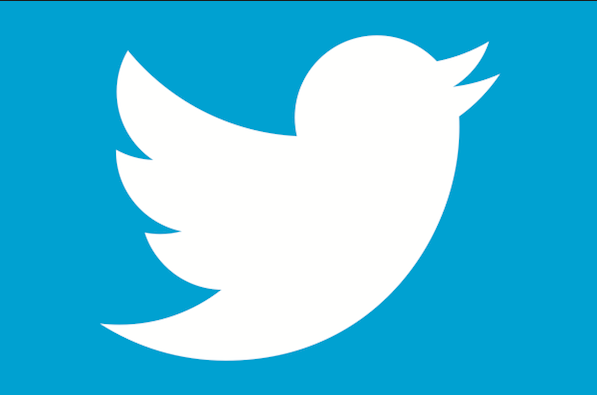


Rapport Projet Encadré

Analyse de Comportements avec Twitter





**PROMOTION : 2017/2018**

**éTUDIANTEs :MA LING, SHA qianqian**

**Git : https://gitlab-etu.fil.univ-lille1.fr/ma/PJB\_SHA\_MA**

SOMMAIRE

[Chapitre 1 Description générale du projet 2](#_Toc501234021)

[1 Description de la problématique 3](#_Toc501234022)

[2 Description générale de l'architecture de l’application 3](#_Toc501234023)

[Chapitre 2 Détails des différents travaux réalisés 4](#_Toc501234024)

[1.API Twitter 4](#_Toc501234025)

[2.Préparation de la base d'apprentissage 5](#_Toc501234026)

[2.1 Nettoyage des données 5](#_Toc501234027)

[2.2 Construction de la base 5](#_Toc501234028)

[3.Algorithme de classification 6](#_Toc501234029)

[3.1 Mots clés 6](#_Toc501234030)

[3.2 KNN 6](#_Toc501234031)

[3.3 Bayes 7](#_Toc501234032)

[4.Interface graphique 8](#_Toc501234033)

[4.1 Copies d'écran 8](#_Toc501234034)

[4.2 Manuel d'utilisation 12](#_Toc501234035)

[Chapitre 3 Résultats de la classification avec le différentes méthodes et analyse 13](#_Toc501234036)

[Chapitre 4 Conclusions 14](#_Toc501234037)

# Chapitre 1 Description générale du projet

Ces dernières années, l'utilisation de réseaux sociaux comme "Twitter" s'est énormément démocratisée. De ce fait il peut être intéressant d'analyser les messages postés ("Tweets") par les utilisateurs afin d'en connaitre leur nature, pour par exemple connaitre leurs sentiments sur un produit, une personne, un concept, ...

## 1 Description de la problématique

L'objectif de ce projet est d'automatiser le traitement des "Tweets" pour en tirer des informations via la création d'une application.

Pour ce faire, nous avons dans un premier temps utilisé l'API Twitter (twitter4j) pour récupérer les informations, autrement dit les tweets des utilisateurs. Dans un second temps nous avons traité les données récupérées pour faire une base d'apprentissage. Puis nous avons trié les tweets à l'aide de différents algorithmes (mots clefs, KNN et Bayésien) de classification et nous avons illustré cette classification avec des graphiques(jfreechart). Pour finir, nous avons réalisé une interface graphique à l'aide de Swing, pour rendre l'application agréable à utiliser.

## 2 Description générale de l'architecture de l’application

Pour réaliser notre application, nous avons choisi d'utiliser le langage Java. Nous l'avons découpé en 4 packages :

* Le package "View" qui contient les classes ayant comme rôle la création de l’interface graphique et le rendu visuel de l’application.
* Le package "Algorithmes" qui contient le code correspondant aux trois algorithmes qui permettent d'analyser la nature des tweets et d'en calculer le taux d'erreurs.
* Le package "Outils" qui contient des fonctions pour aider à la création des fonctions des autres packages ainsi que pour construire des diagrammes. Comme son nom l'indique ce package est une "boite à outils".
* Le dossier "Ressource" qui contient les images et les fichiers utiles à la réalisation du projet.

../../../../../Desktop/核心编码树状图.pdf

Architecture de notre application

# Chapitre 2 Détails des différents travaux réalisés

## 1.API Twitter

Dans ce projet, nous avons dû utiliser une interface de programmation (API) de Twitter pour obtenir les tweets par mot-clé. Pour ce faire, on s'est inscrit sur le site "<https://developer.twitter.com/>" pour obtenir le droit d'utiliser l'API. Puis on a téléchargé cet API en ligne et on l'a utilisé dans la "Library" de notre application. Dans nos codes, on a configuré les informations (API keys, API secret, access token et access token secret) et utiliser le "TwitterFactory" pour obtenir le permis d'accès aux informations de twitter. A l'aide de la classe "Query", on peut chercher les tweets récents par mot-clé, limiter la quantité de tweets d'une recherche et la langue (fr) des tweets. La classes "Status" nous permets de nous aider à obtenir toutes les informations relatives : ID du tweet, l’utilisateur, etc.

## 2.Préparation de la base d'apprentissage

Pour préparer la base d'apprentissage, on a récupéré 300 tweets, que l'on a classifié (positif=2, négatif=0 ou neutre=1) à la main. Puis nous avons "nettoyé" ces tweets, c'est à dire que l'on a supprimé tous les caractères inutiles (par exemple les smileys). Nous avons conservé ces tweets dans un fichier au format CSV(twitter\_annotation.csv).

### 2.1 Nettoyage des données

Pour nettoyer les tweets, nous avons procédé en 3 étapes :

* La première pour filtrer et supprimer tous les caractères spéciaux correspondant à des smileys (ou emoji, émoticône, …), grâce à notre fonction *"filter\_contenu"* du package outils.
* La deuxième pour supprimer les liens, les re-tweets, les espaces consécutifs, les @ ainsi que les #, grâce à notre fonction *"pattern\_contenu"* du package outils.
* La troisième pour supprimer les tweets doublons, en analysant lorsqu'on veut ajouter un tweet dans notre base s'il existe déjà ou non.

Une fois ces trois étapes exécutées grâce à des opérations Java sur les String, nous avons pu construire la base.

### 2.2 Construction de la base

Nous construisons une base dans un fichier CSV de 300 tweets après nettoyage avec différents sujet (vélo, neige, enfant, film...) pour lesquels nous avons mis l'annotation. Par exemple, dans notre page principale, si on appuie le bouton *"Ajoute-annotation"*, on peut donner manuellement les annotations pour les tweets que l'on veut ajouter dans la base des tweets.



Exemple d'ajout d'annotations

## 3.Algorithme de classification

Pour analyser les tweets, nous utilisons 3 différents algorithmes de classification : la méthode dictionnaires, la méthode KNN et enfin la méthode Bayes.

Dans notre package *"algo"*, il y a trois classes correspondantes aux trois algorithmes. Et sur la page principale de notre application, il y a un menu *"algo"* où on peut choisir quel algorithme de classification utiliser. Son résultat va s'afficher dans une autre fenêtre.

### 3.1 Mots clés

La méthode des mots clés consiste à trouver la nature d'un tweet grâce à deux fichiers de listes de mots (positifs et négatifs). On compare chaque mot des nouveaux tweets avec les deux listes de mots. Pour chaque tweet, s'il contient plus de mots positifs que négatifs, alors ce tweet est positif, et réciproquement pour les tweets négatifs. S'il contient autant de mots positifs que négatifs ou s'il ne contient pas de mots des listes, alors il est neutre.

Pour se faire, nous appelons la fonction *"analyseTheMessage( )"* qui prend un tweet en paramètre et qui calcule le ratio positif/négatif du tweet par la méthode des mots clés, en comparant chaque mot de ce tweet avec nos listes de mots positifs et négatifs. Elle retourne l'annotation de ce tweet. Nous avons également créé une fonction *"algo\_Mot\_cle( )"*, qui permet d'utiliser cette méthode pour analyser les 20 tweets qui apparaissent après une recherche sur notre application. Cette fonction prend une liste de tweets en paramètre et retourne la liste des annotations. Pour afficher ces résultats dans notre application nous avons créé la classe *"Mot\_cle\_frame"*.

### 3.2 KNN

L’algorithme k-nearest neighbors ou KNN (k voisins les plus proches), est un algorithme qui consiste à prendre en compte les k voisins les plus proches, de la base d’apprentissage, pour notre nouvelle entrée. On peut ainsi déterminer la valeur la plus probable pour cette entrée en fonction des k voisins. Pour notre programme, cela s’applique de la manière suivante : Pour un nouveau tweet, on calcule sa distance (nombre de mots en communs) avec tous les tweets de notre base d’apprentissage.

Pour se faire, nous appelons la fonction*"distance(t1,t2)"* pour calculer la distance entre deux tweets *(Distance(t1, t2)=(nombre\_total\_de\_mots – nombre\_de\_mots\_commun)/nombre\_total de mots)*., et la fonction *"Tous\_les\_distances (t, liste\_t)"* qui prend un tweet en paramètre et calcule sa distance avec tous les tweets dans la base d’apprentissage. Puis on choisit les 5 tweets de la base les plus proches du nouveau et on annote le nouveau tweet en fonction de la majorité d’annotation (s'il y a plus de tweet positif dans notre échantillon de 5 tweets alors notre nouveau tweet est positif, même raisonnement pour négatif ou neutre). Nous avons également créé une fonction *"algo\_knn (liste\_nouveaux\_tweets, liste\_base\_tweets)"*, qui permet d'utiliser cette méthode pour analyser les 20 tweets qui apparaissent après une recherche de tweets sur notre application. Cette fonction prend une liste de tweets en paramètre et retourne la liste des annotations. Pour afficher ces résultats dans notre application nous avons créé la classe *"Knn\_frame"*.

### 3.3 Bayes

#### 3.3.1 Représentation de présence

L’algorithme de classification bayésienne est un outil de classification qui s’exprime par : *p(m|c)=n(m,c)/n(c))* avec, dans notre cas, c une classe (positif, négatif ou neutre), m un mot, n(c) le nombre total de tweets de la classe c, n(m, c) le nombre tweets qui contient le mot m dans la classe c et P(m|c) la probabilité de m sachant c et p(c) .le ratio de tweets de classe c dans tous les tweets de la base.

Pour connaitre la classe c d’un tweet, on le divise en mot dans un premier temps, dans un second temps on cherche le nombre d’occurrence des tweets qui contiennent ce mot m. Puis on multiple les occurrences de tous les mots par la proportionnalité de tweets de classe c de la base de tweets.

Si le mot n’apparait jamais dans la classe c, on remplace le résultat 0 par (p(m|n)+1)/n(c)+N. Ensuite on multiple le résultat avec p(c), pour obtenir la probabilité que ce tweet apparaisse dans la classes c. En Comparant avec les probabilités des trois classes, on peut annoter ce tweet par la classes qui a la probabilité la plus grande.

Pour se faire, nous appelons la fonction *"AlgoUnigramme (String mot, List<Bayes> classes)"* qui calcule la probabilité de p(m|c), et la fonction *"AlgoUnigramme\_res(String[] mots\_twitter)"* qui calcule la probabilité de p(c|t). Nous avons également créé une fonction *"algoBayes(String moodle, Bayes b)"*, qui permet d'utiliser cette méthode pour analyser les 20 tweets qui apparaissent après une recherche de tweets sur notre application. Cette fonction prend une liste de tweets en paramètre et retourne la liste des annotations. Pour afficher ces résultats dans notre application nous avons créé la classe *"Bayes\_frame"*.

#### 3.3.2 Représentation de fréquence

Dans ce cas, on prend en compte le nombre de mots m dans le nouveau tweet t (nm), on met la probabilité P(m|c) à la puissance (nm). Et on crée aussi une fonction pour nettoyer les tweets des mots de moins de trois lettres qu'on appelle *"filter\_moins\_de\_trois\_lettre(Bayes bn)"*.

#### 3.3.3 Ensembles de mots

Parce que certaines combinaisons de mots peuvent avoir de l'importance, on utilise trois façons de comparer les mots.

* **Uni-Gramme**

Le uni-gramme est défini par un mot simple, par exemple la phrase : *j'aime la pomme* contient 3 uni-grammes : *j'aime, la, pomme.*

* **Bi-Gramme**

Le bi-gramme est considéré deux mots consécutifs, par exemple la phrase : *j'aime la pomme,* contient 2 bi-grammes: (j'aime la), (la pomme).

* **Uni-Bi-Gramme**

Le uni-bi-gramme est la combinaison des uni-grammes + bi-grammes, par exemple la phrase : *j'aime la pomme.* contient 5 uni-bi-grammes: *j'aime, la, pomme, (j'aime la), (la pomme).*

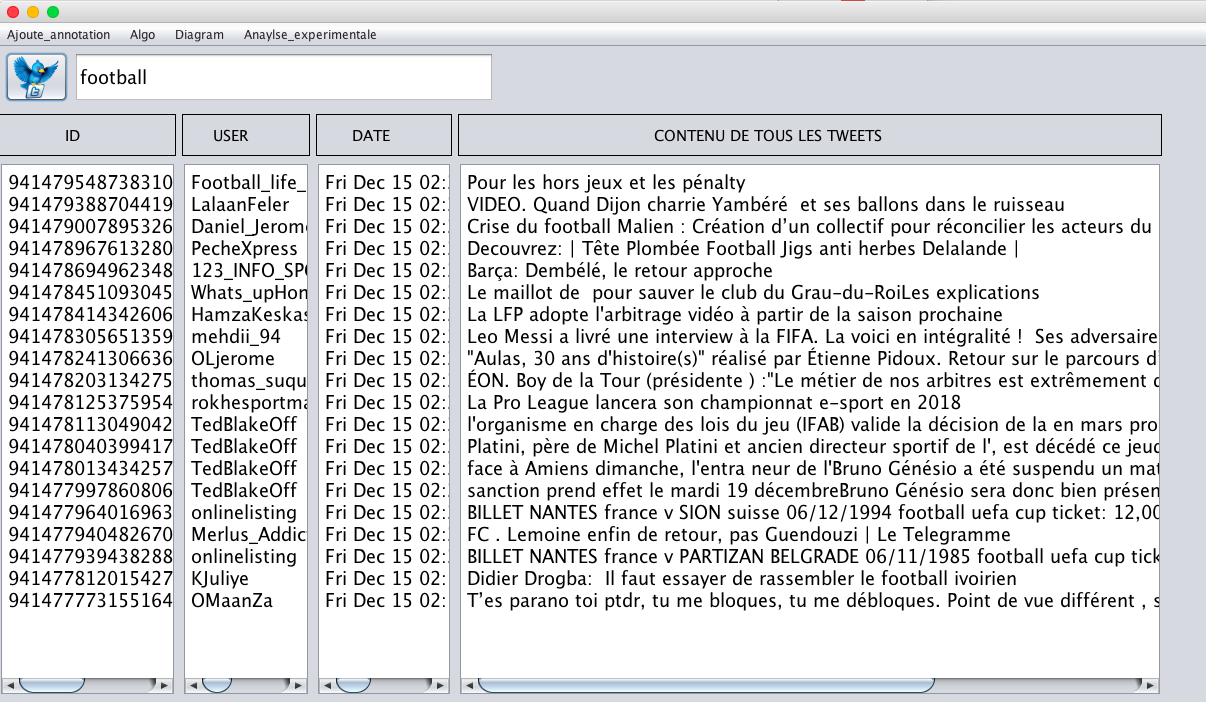
## 4.Interface graphique

Notre interface graphique se compose d'un écran principal depuis lequel on peut accéder à plusieurs autres écrans reflétant des fonctionnalités que l'on a implémenté :

### 4.1 Copies d'écran

Vous trouverez ci-dessous l’écran principal de notre application. Il a 3 parties :

* Menu : permettant de choisir différente fonction
* Champ de recherche : ou on peut entrer le mot-clé que l'on veut chercher.
* Tableau d'affichage : afficher des tweets séparés en 6 parties : ID, nom de d'utilisateur, date, contenu de tweets



Page principale

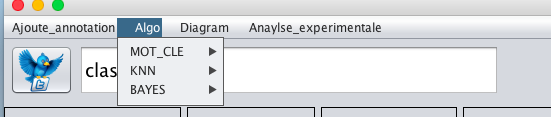
De plus, nous créons des sous-fenêtres contrôlées par les menus de la page principale.

1.Ajoute-annotation (On peut donner les annotations aux nouveaux tweets à la main)

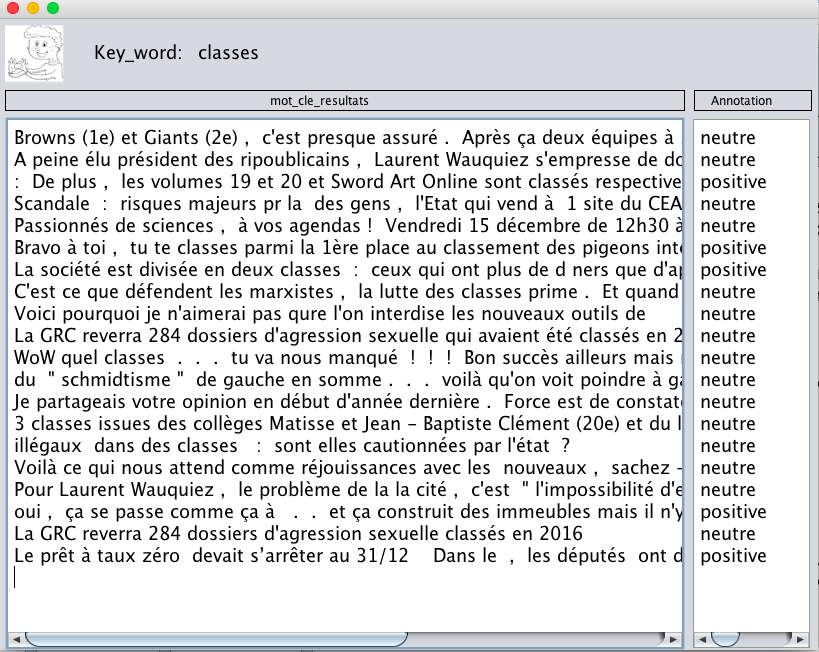


Ajouter les annotations

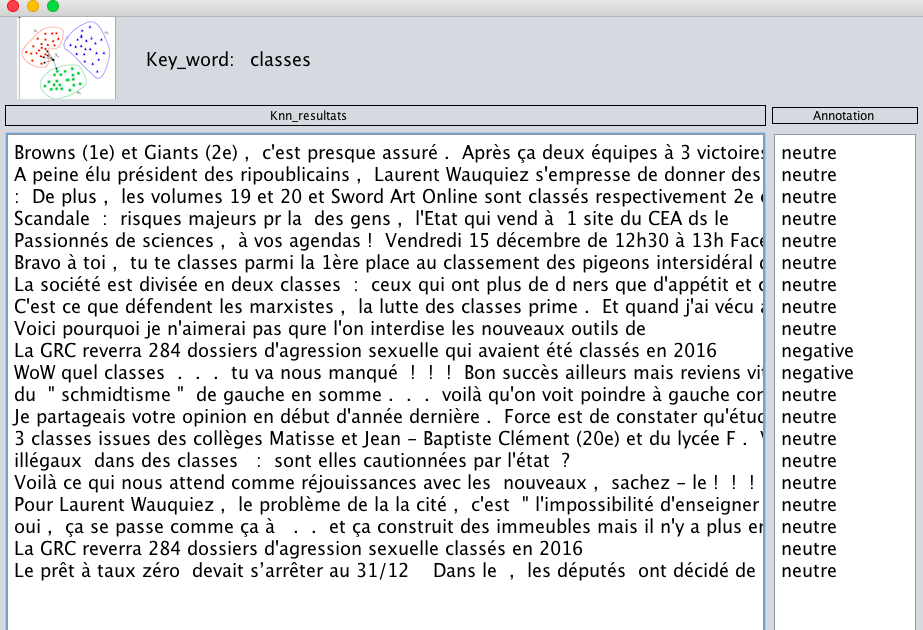
2.Algo (Utiliser différents algorithmes pour annoter les nouveaux 20 tweets)



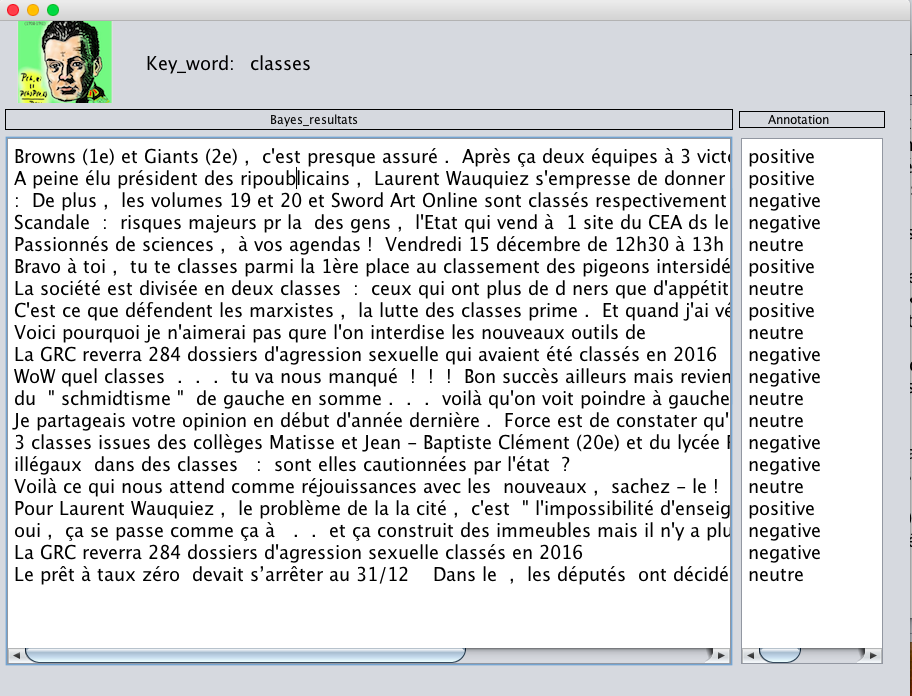
Le menu des algorithmes



Mot-clé

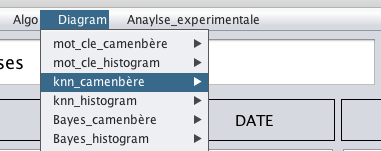


Knn

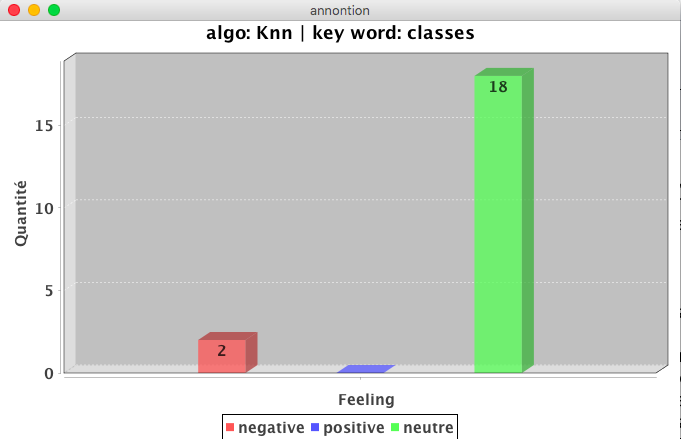


Bayes

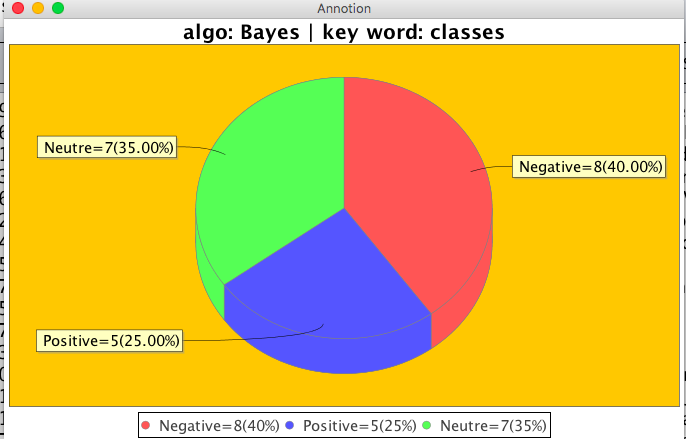
3. Diagram (Permet de mettre en valeur les résultats sous forme de diagrammes)



Le menu de diagrammes

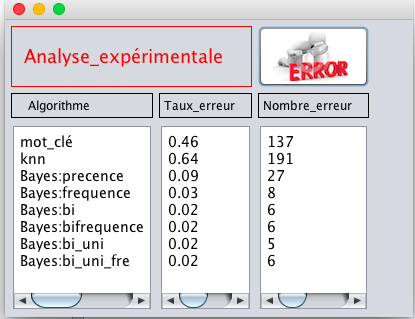
****

**L'histogramme**

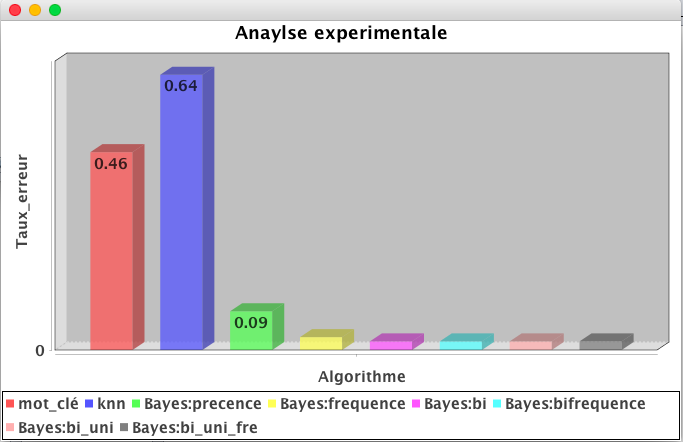
****

**Le camembert**

4. Analyse\_experimentale (Permet d'afficher les taux d'erreurs de tous les algorithme)



Le taux d'erreurs de tous les algorithmes



Le diagramme d'erreurs

### 4.2 Manuel d'utilisation

**4.2.1 Exécuter l'application :**

Pour exécuter l'application il suffit de double cliquer sur *"pjb\_twitter.jar".* L'environnement de java doit être java 7 ou supérieur.

#### 4.2.2 Chercher un mot clef :

Entrez un mot clef dans la barre de recherche, et puis cliquez sur le bouton avec le petit oiseau pour rechercher les tweets. Le résultat des tweets correspondant sera affiché.

#### 4.2.3 Faire une annotation manuelle et l'ajouter dans la base des tweets :

Cliquer sur l'onglet "Ajoute\_annotation", dans la barre de menu, pour annoter manuellement les tweets et cliquer sur "OK" pour valider. Les tweets avec annotations manuels seront sauvegardés dans le fichier “twitter\_annotation.csv”.

#### 4.2.4 Faire une annotation automatique via différents algorithmes :

Cliquer sur l'onglet "Algo ", dans la barre de menu, pour ouvrir le menu déroulant permettant de sélectionner l'algorithme avec lequel on souhaite faire l'annotation automatique.

Une fois choisi, le programme ouvre une nouvelle fenêtre affichant les résultats

#### 4.2.5 Faire un diagramme automatique via différents algorithmes :

Cliquer sur l'onglet "Diagram ", dans la barre de menu, pour ouvrir le menu déroulant permettant de sélectionner l'algorithme avec lequel on souhaite faire dessiner son digramme ainsi que le style de diagramme (camembert, histogramme).

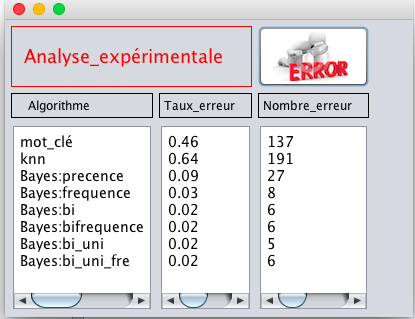
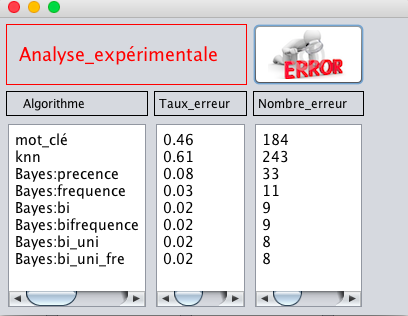
Une fois choisi, le programme ouvre une nouvelle fenêtre affichant le diagramme.

#### 4.2.6 Calculer les taux d'erreurs de tous les algorithmes :

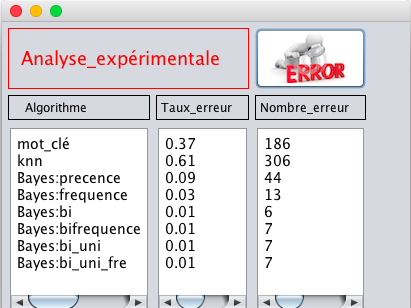
Cliquer sur l'onglet "Analyse expérimentale ", dans la barre de menu, pour ouvrir le menu analyse expérimentale permettant de voir les taux d'erreurs des différents algorithmes.

Dans ce menu double cliqué sur le bouton "error" pour afficher les statistiques sous forme de diagramme.

# Chapitre 3 Résultats de la classification avec le différentes méthodes et analyse

300 tweets 400 tweets



500 tweets

Dans notre base de tweets, il y a 3 situations ,300 tweets,400 tweets et 500 tweets. On calcule le taux d'erreurs par l'analyse expérimentale, d'après nos résultats, pour :

La méthode mot clé, il y a un taux d'erreurs d'environ 40%.

La méthode Knn, il y a un taux d'erreurs d'environ 60%.

Les méthodes bayésiennes sur la présence en unigramme a un taux d'erreurs d'environ 9%.

Les méthodes bayésiennes sur la fréquence en unigramme a un taux d'erreurs d'environ 3%.

Les méthodes bayésiennes sur la présence en bigrammes a un taux d'erreurs d'environ 2%.

Les méthodes bayésiennes sur la fréquence en bigrammes a un taux d'erreurs d'environ 2%.

Les méthodes bayésiennes sur la présence en uni+bi a un taux d'erreurs d'environ 2%.

Les méthodes bayésiennes sur la fréquence en uni+bi a 5 erreurs, le taux d’erreur est 2%.

Grâce à ces résultats, on peut en déduire que les méthodes de Bayes sont les plus efficaces et parmi les méthodes de Bayes, les méthodes en uni+bi sont les meilleurs.

# Chapitre 4 Conclusions

Ce projet nous a permis de concevoir une application fonctionnelle, qui permet d'analyser des tweets en fonction d'un mot clé, de les trier automatiquement par leur nature (positive, négative, neutre) via différents algorithmes (knn, bayes, mots clés), de représenter les résultats sous forme de graphes et de calculer les taux d'erreurs des différents algorithmes pour vérifier la fiabilité des résultats.

Dans ce projet, nous avons appris à utiliser des APIs pour obtenir leurs fonctionnalités et nous en servir pour notre projet. De plus nous avons appris de nouveaux algorithmes de classification.

Nous avons utilisé NetBean et swing pour la première fois, nous permettant de créer des interfaces graphiques facilement. La méthode de travail était agréable, en effet nous avons apprécié l'autonomie de travail tout en aillant des professeurs à l'écoute pour nous guider quand nous en avions besoin.

Cependant nous avons rencontré de nombreux problèmes, au niveau du design de l'application, en effet nous n'étions jamais vraiment satisfaites de notre interface. De plus nous avons eu des difficultés à retranscrire les algorithmes de classification sous forme de code.

Pour finir, je pense que l'on pourrait améliorer notre application, en permettant d'analyser tous les tweets postés par un utilisateur, pour analyser son avis général. Cela permettrait aussi de connaître l'évolution de ses pensées sur différents sujets.