

PJE : Analyse de Comportements avec Twitter

Belkadi Khaled — Ouartsi Wassim

Décembre 2015

Contents

1	Introduction	3
1.1	Background	3
1.2	Sommaire	3
2	Implémentation du système	4
2.1	Technologie Review	4
2.1.1	JavaFX	4
2.1.2	SQLite	4
2.1.3	Twitter4J	5
2.2	Architecture Logicielle	6
3	Algorithmes de classification	7
3.1	Dictionnaire (mot-clès)	7
3.2	K-Nearest Neighbors	8
3.3	Bayésien	9
3.3.1	Bayésien par présence	9
3.3.2	Bayésien par fréquence	10
4	Analyse expérimentale	11
5	Présentation de l'interface	13
5.1	L'onglet de recherche	13
5.2	L'onglet base de données	14
5.3	L'onglet statistiques	15
5.4	L'onglet paramètres	16
5.5	L'onglet analyse	17
6	Conclusion	18
6.1	Objectifs	18
6.2	Futures améliorations	18

1 Introduction

1.1 Background

Tweet Analyser est un projet réalisé dans un cadre scolaire, il consiste en la conception d'une application permettant d'émettre un sentiment (négatif, neutre, positif) à des tweets, vis-à vis d'une idée, d'un concept ou d'une orientation sociale, d'un personnage politique, d'un produit commercial, technologique, etc.

Premièrement, l'application devra être capable de récolter des tweets. Ensuite de les formater (nettoyer), afin d'obtenir qu'une chaîne de mots, tout en omettant les hashtags, ponctuations, liens, etc.

Deuxièmement, les tweets récoltés devront être soumis à des algorithmes de classification (dictionnaire, KNN, bayes...), afin d'extraire automatiquement un sentiment général et par tweet.

Finalement, l'application développée devra satisfaire les interactions de l'utilisateur à l'aide d'une interface graphique attrayante tant au niveau ergonomique qu'au niveau du "look and feel".

1.2 Sommaire

Ce document commence avec la présentation des choix technologiques, l'architecture logicielle. Ensuite, vient l'analyse des algorithmes de classification de l'application, soit : dictionnaire (mot-clés), K-nearest neighbours, Bayésien simple/fréquence n-grammes.

2 Implémentation du système

2.1 Technologie Review

2.1.1 JavaFX

Lors de l'énoncé du projet, il était proposé de développer l'interface graphique à travers SWING, cependant nous n'avons pas utilisé cette technologie, car celle-ci est en phase de devenir dépréciée. C'est pourquoi, nous avons opté pour JavaFx, qui est un ensemble de composants graphiques et médias permettant aux développeurs de concevoir, créer, tester, déboguer et déployer des applications clients (front-end *rightarrow* view), ainsi qu'un middleware (ViewController).

Cette technologie a comme avantages l'utilisation du langage JavaFX FXML (basé sur le concept XML), celui-ci fournit des structures ayant pour but de construire une interface utilisateur, tout en omettant la logique du code applicatif.

2.1.2 SQLite

Nous avons délaissé l'idée d'organiser notre base de données via CSV, au profit de SQLite. Car, celui-ci propose un moteur de base de données relationnelle accessible par le langage SQL. De plus, contrairement aux serveurs de bases de données traditionnels, comme MySQL ou PostgreSQL, SQLite a la particularité de ne pas reproduire le schéma habituel client-serveur mais d'être directement intégrée aux programmes. En d'autres termes, l'intégralité de la base de données (déclarations, tables, index et données) est stockée dans un fichier indépendant de la plateforme.

Voici les raisons qui nous ont poussé à privilégier SQLite face à CSV :

1. SQLite comporte des algorithmes d'indexation, aidant les requêtes de recherche.
2. Il n'est pas nécessaire d'effectuer des lectures/écritures de façon manuelle, normalisation à l'aide du langage SQL.
3. Du fait du point 2, il n'est plus nécessaire d'effectuer des explodes/implodes des lectures/écritures.

2.1.3 Twitter4J

Twitter4J est une bibliothèque Java non-officielle pour l'API Twitter. Elle permet facilement d'intégrer une application Java avec les services Twitter.

Twitter4J est caractérisée par :

- Entièrement faite en Java.
- Fonctionne sur toutes les plateformes Java a partir de la version 5.
- Compatible avec Android et Google App Engine.
- Ne dépend d'aucune autre librairie.
- Compatible a cent-pour-cent avec l'API Twitter 1.1.

2.2 Architecture Logicielle

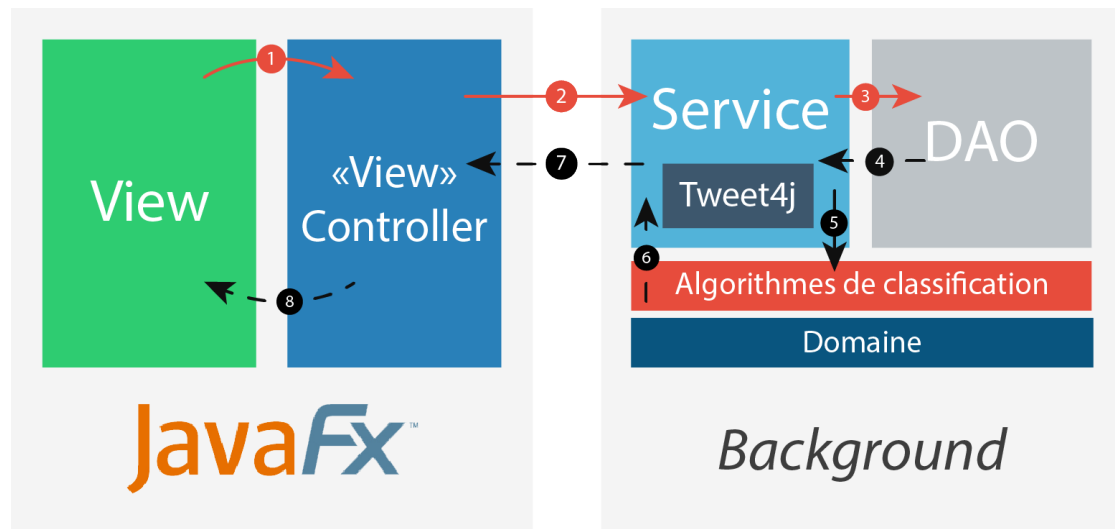


Figure 1: Architecture de l'application Tweet Analyser

Exemple de l'architecture lors d'un appel d'un algorithme.

1. L'utilisateur demande un traitement sur un algorithme x.
2. La ViewController est le middleware entre le front-end et le back-end. Celui-ci va transmettre les tweets à traiter par l'algorithme x.
3. Le service récupère les données nécessaires en database à travers une instance de la DAO (data access object) pour l'algorithme x.
4. La DAO retourne les données au Service.
5. Le service transfère : les données récupérées par la DAO, ainsi que la liste des tweets à traiter par l'algorithme x.
6. L'algorithme renvoie la liste des tweets livrés par l'algorithme.
7. Le service retourne la liste des tweets traités à la ViewController.
8. La ViewController rafraîchit l'observable liste des tweets, ensuite génère les graphiques.

3 Algorithmes de classification

”On appelle classification automatique la catégorisation algorithmique d’objets. Elle consiste à attribuer une classe ou catégorie à chaque objet (ou individu) à classer, en se basant sur des données statistiques. Elle fait couramment appel à l’apprentissage automatique et est largement utilisée en reconnaissance de formes.”

ref - Wikipédia

3.1 Dictionnaire (mot-clès)

Le premier algorithme que nous avons implémenté est le celui basé sur des dictionnaires. Celui-ci consiste à classer un Tweet en dénombrant le total de mots positifs, ou négatifs contenu dans ce dernier.

Nous avons utilisé deux dictionnaires, le premier contenant un ensemble de 2010 mots positifs, et le second un ensemble de mots négatifs.

1. L’algorithme reçoit en entrée une liste de Tweets à annoter.
2. L’algorithme divise chaque Tweet en un ensemble de mots.
3. Pour chaque mot vérifie sa présence dans le dictionnaire positif.
4. Si Vrai il incrémente la variable *delta*.
5. Sinon il vérifie la présence du mot dans le dictionnaire négatif.
6. Si Vrai il décrémente la variable *delta*.
7. Enfin l’algorithme vérifie la valeur de *delta* :
 - Si elle est positive le Tweet est annoté positif.
 - Sinon le Tweet est annoté négatif.
 - Et dans le cas où $delta = 0$ le Tweet est considéré neutre.

3.2 K-Nearest Neighbors

L'algorithme de KNN se base sur le calcul d'une distance entre une nouvelle entrée *Tweet* et ses K plus proches voisins.

Dans notre implémentation de KNN, nous avons utilisé la distance euclidienne.

1. L'algorithme reçoit en entrée une liste de Tweets à annoter.
2. L'algorithme récupère la liste des Tweets en base de données.
3. Pour chaque Tweet, l'algorithme va calculer la distance entre ce dernier et chacun de ses K plus proches voisins.
 - Diviser le *Tweet* en un ensemble de mots.
 - Diviser chaque voisin en un ensemble de mots.
 - Calculer le nombre de mots communs entre le *Tweet* et chacun de ses voisins.
 - On obtient la distance par la formule suivante :

$$1 - \frac{nbMotsCommuns}{nbMotsTweet + nbMotsVoisinCourant}$$

- Ajouter la distance calculée dans une liste regroupant les distances entre le *Tweet* et ses voisins.
 - Répéter jusqu'à terminer tous les voisins.
 - On récupère l'annotation du Tweet ayant la plus petite distance dans la liste et on l'affecte au nouveau *Tweet*.
4. Répéter jusqu'à terminer tous les *Tweets*

En ce qui concerne la version mixte la logique ne change pas mais on prend en compte toutes les probabilités.

3.3 Bayésien

La première étape nécessaire afin d'implémenter la classification bayésienne, consiste en la mise à jour de la liste des vocabulaires disponibles en base de données à chaque nouvelle entrée de tweet dans celle-ci. Un vocabulaire est composé de : une chaîne de n-gram mot, son nombre d'occurrences dans les Tweets positives, négatives ou neutres.

3.3.1 Bayésien par présence

1. L'algorithme reçoit en entrée une liste de Tweets à annoter, ainsi que le nombre de mots à utiliser (n-grams)
2. Le système récupère la liste des vocabulaires générés précédemment.
3. Pour chaque Tweet on récupère tous les mots.
 - Pour chaque mot dans le vocabulaire :

$$P_{Pos|Neg|Neu} \times = \frac{nbOccurrences(Pos|Neg|Neu) + 1}{nbTotalMots(Pos|Neg|Neu) + nbTotalMotsVocabulaire}$$

- Répéter pour tous les mots du Tweet.
4. On calcule par la suite :

$$P_{Pos|Neg|Neu} \times = \frac{nbTotalMots(Pos|Neg|Neu)}{nbTotalMotsVocabulaire}$$

5. Et selon le max entre P_{Pos} , P_{Neg} , P_{Neu} on annote le Tweet courant.
6. Répéter pour tous les Tweets.

En ce qui concerne la version mixte la logique ne change pas mais on effectue le calcul jusqu'au n^{eme} -gramme.

3.3.2 Bayésien par fréquence

1. L'algorithme reçoit en entrée une liste de Tweets à annoter, ainsi que le niveau de gramme demandé.
2. Le système récupère la liste des vocabulaires générés précédemment.
3. Pour chaque Tweet on récupère tous les mots.
 - Pour chaque mot on récupère sa fréquence dans le Tweet.
 - Pour chaque mot dans le vocabulaire :

$$P_{Pos|Neg|Neu} \times = \frac{nbOccurrences(Pos|Neg|Neu) + 1}{nbTotalMots(Pos|Neg|Neu) + nbTotalMotsVocabulaire}$$

$$P_{Pos|Neg|Neu} = P_{Pos|Neg|Neu}^{(frequencyDuMot)}$$

- Répéter pour tous les mots du Tweet.
4. Et selon le max entre $P_{Pos}, P_{Neg}, P_{Neu}$ on annote le Tweet courant.
 5. Répéter pour tous les Tweets.

En ce qui concerne la version mixte la logique ne change pas mais on effectue le calcul jusqu'au n^{eme} -gramme.

4 Analyse expérimentale

Lors de nos différents tests avec la méthode de cross-validation on a très vite remarqué que l'algorithme avec dictionnaires offrait les plus piètres résultats, et que Bayès par fréquences bi-grammes offrait les meilleurs.

Voici les résultats obtenus avec plusieurs valeurs pour KNN.

Nous avons essentiellement utilisé la racine carrée car d'après nos recherches, une des meilleurs valeurs à utiliser pour le nombre de voisins est la racine carrée de tout l'ensemble.

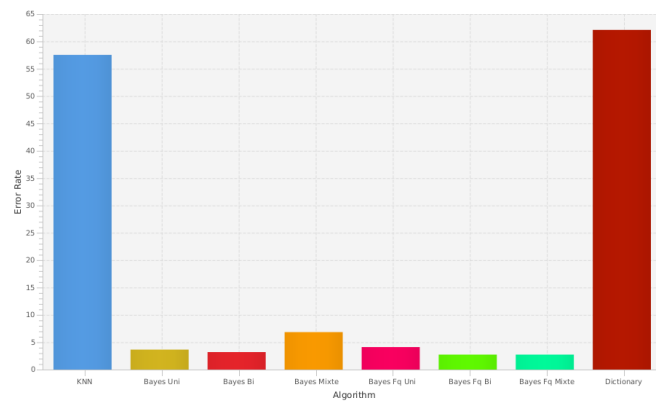


Figure 2: Pour KNN : La racine carrée de l'ensemble a annoter.

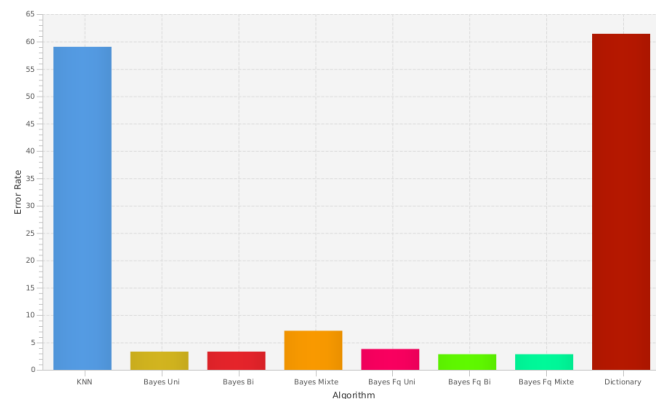


Figure 3: Pour KNN : La racine carrée du nombre de tous les Tweets en BDD.

On remarque les meilleurs résultats sont obtenus avec comme valeur pour K la racine carrée de l'ensemble a annoter.

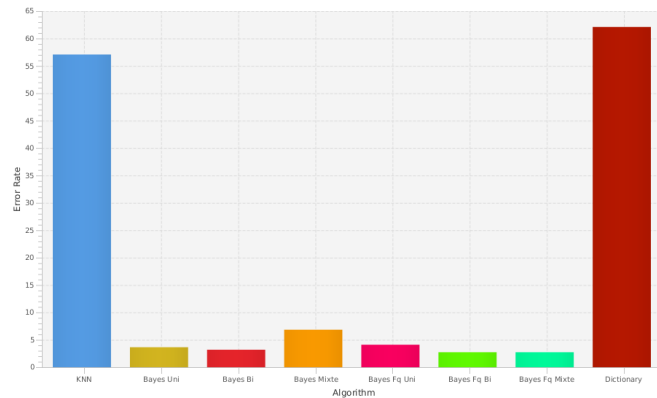


Figure 4: Pour KNN : La racine carrée de l'ensemble d'apprentissage.

D'après les résultats obtenus on pourrait effectuer le classement des algorithmes selon leur taux d'erreurs comme suit :

1. Bayes par fréquences bi-grammes.
2. Bayes par fréquences mixte.
3. Bayes par fréquences.
4. Bayes naïve bi-grammes.
5. Bayes naïve.
6. Bayes mixte.
7. KNN
8. Dictionnaire.

De ce fait, et selon nos expérimentations et notre base d'apprentissage contenant plus de 210 Tweets, nous pouvons dire que la méthode de classification " Bayes par fréquences bi-grammes " est la meilleure.

5 Présentation de l'interface

5.1 L'onglet de recherche

Il suffit d'entrer son mot-clé dans le champ recherche, et de choisir un algorithme d'annotation, le configurer si nécessaire, et lancer la recherche.

Une fois les Tweets affichés, 3 types de smileys illustrent l'annotation attribué à chaque Tweet.

- Bleu : Positif.
- Jaune : Neutre.
- Rouge : Négatif.

Il est possible de voir la tendance des Tweets recherchés en appuyant sur le bouton en bas à droite.

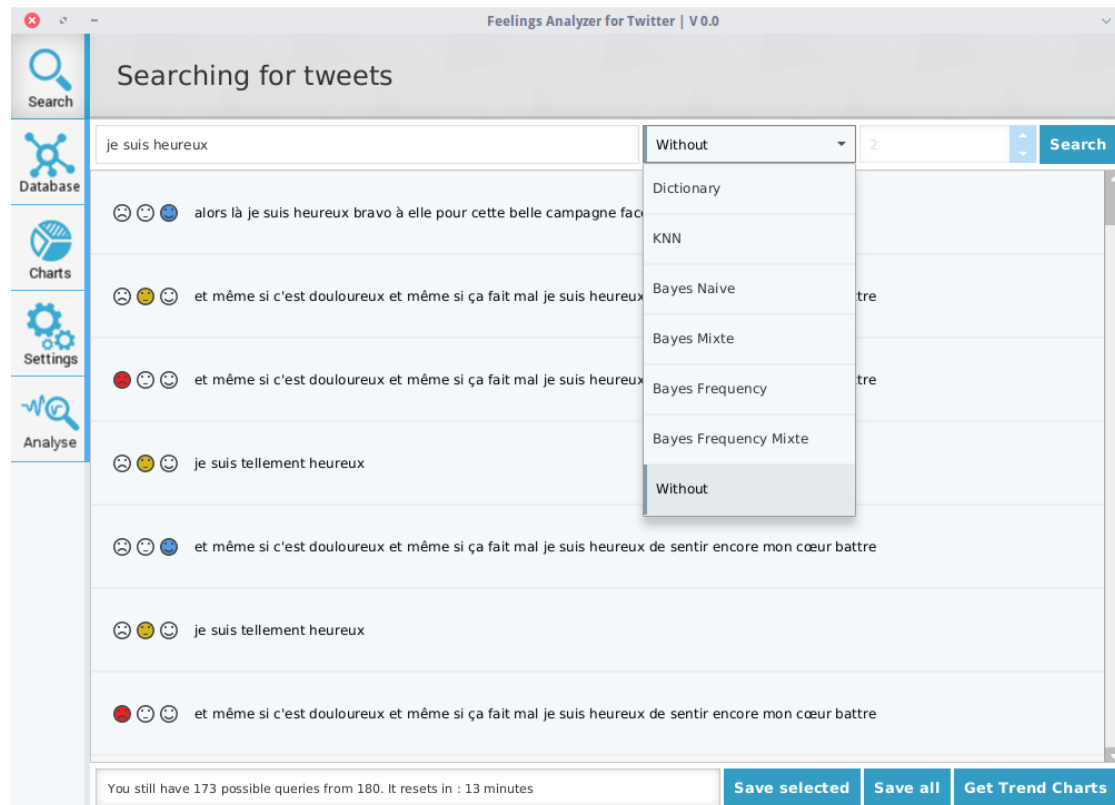
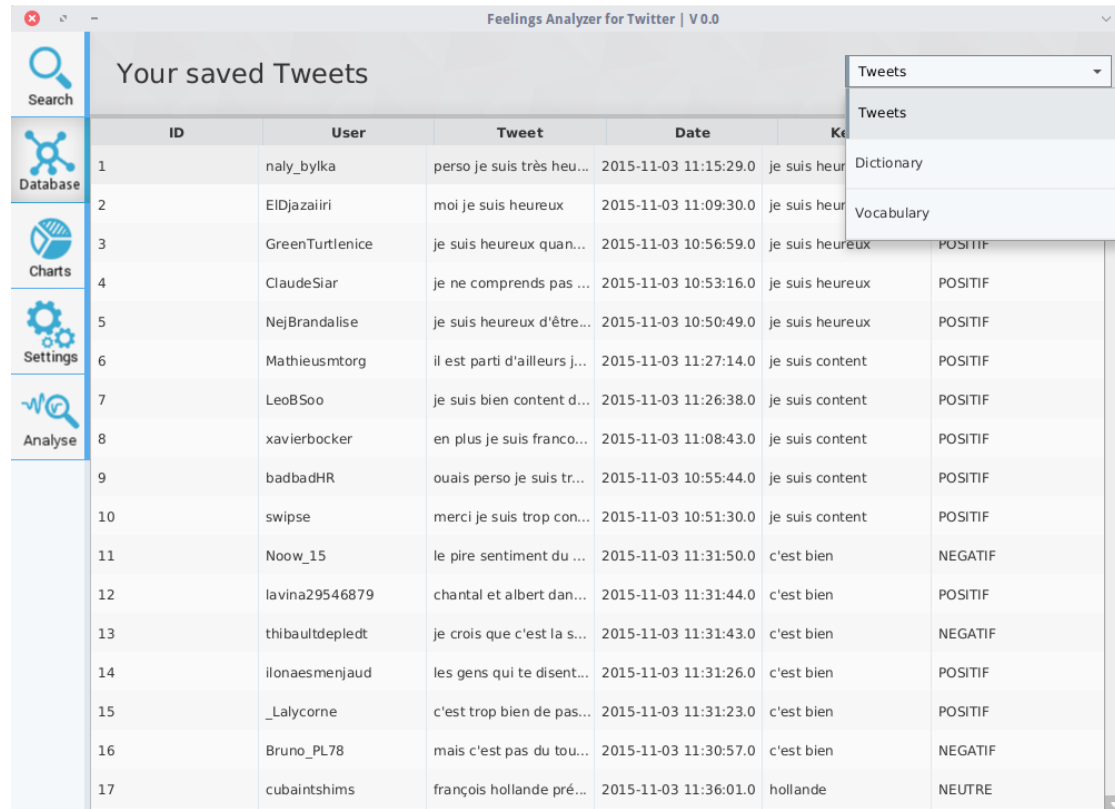


Figure 5: L'écran de recherche.

5.2 L'onglet base de données

Il permet de visualiser les Tweets sauvegardés en base de données, mais aussi les mots des dictionnaires et ceux du vocabulaire.



ID	User	Tweet	Date	Keywords	Sentiment
1	naly_bylka	perso je suis très heu...	2015-11-03 11:15:29.0	je suis heur	
2	EIDjazairi	moi je suis heureux	2015-11-03 11:09:30.0	je suis heur	
3	GreenTurtlenice	je suis heureux quan...	2015-11-03 10:56:59.0	je suis heureux	POSITIF
4	ClaudeSiar	je ne comprends pas ...	2015-11-03 10:53:16.0	je suis heureux	POSITIF
5	NejBrandalise	je suis heureux d'être...	2015-11-03 10:50:49.0	je suis heureux	POSITIF
6	Mathieusmtorg	il est parti d'ailleurs j...	2015-11-03 11:27:14.0	je suis content	POSITIF
7	LeoBSoo	je suis bien content d...	2015-11-03 11:26:38.0	je suis content	POSITIF
8	xavierbocker	en plus je suis franco...	2015-11-03 11:08:43.0	je suis content	POSITIF
9	badbadHR	ouais perso je suis tr...	2015-11-03 10:55:44.0	je suis content	POSITIF
10	swipse	merci je suis trop con...	2015-11-03 10:51:30.0	je suis content	POSITIF
11	Noow_15	le pire sentiment du ...	2015-11-03 11:31:50.0	c'est bien	NEGATIF
12	lavina29546879	chantal et albert dan...	2015-11-03 11:31:44.0	c'est bien	POSITIF
13	thibaultdepledt	je crois que c'est la s...	2015-11-03 11:31:43.0	c'est bien	NEGATIF
14	ilonaesmenjaud	les gens qui te disent...	2015-11-03 11:31:26.0	c'est bien	POSITIF
15	_Lalycorne	c'est trop bien de pas...	2015-11-03 11:31:23.0	c'est bien	POSITIF
16	Bruno_PL78	mais c'est pas du tou...	2015-11-03 11:30:57.0	c'est bien	NEGATIF
17	cubaintshims	françois hollande pré...	2015-11-03 11:36:01.0	hollande	NEUTRE

Figure 6: L'écran base de données.

5.3 L'onglet statistiques

Il permet de voir la tendance générale des Tweets sauvegardés, mais aussi de Tweets particuliers en faisant une recherche par mots clés sur le champ en haut.

Deux types de représentations ont été implémentées :

- Représentation en Camembert.
- Représentation en Histogramme.

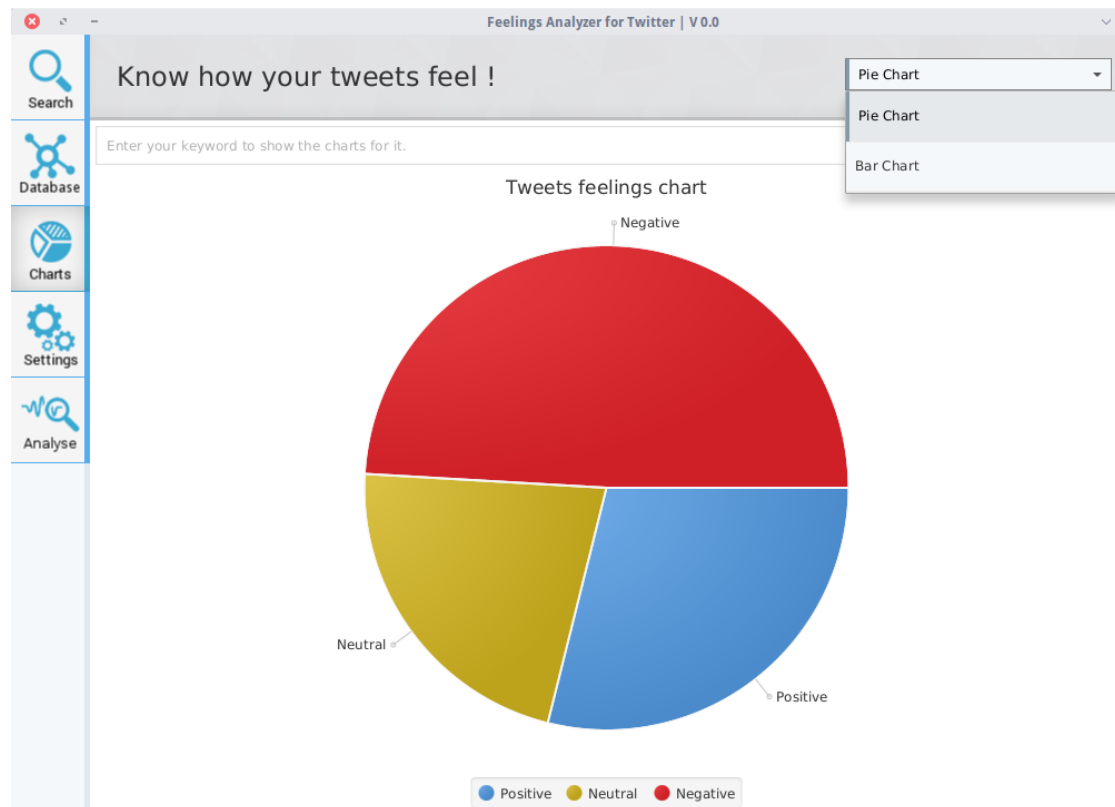


Figure 7: L'écran statistiques.

5.4 L'onglet paramètres

Il permet de configurer l'application :

- Pour qu'elle marche avec vos informations Twitter.
- Vous pourrez configurer un proxy.
- Ajouter vos propres dictionnaires de mots.

Feelings Analyzer for Twitter | V 0.0

Editing settings

Authentication Settings

Consumer Key	<input type="text" value="xTG8A6uRZ189LN0JbsOSQITXK"/>
Consumer Key Secret	<input type="text" value="BAayLVVyMwbPHrZaguqSaPigwcSfvU3sJSROXqXhERakhRSHjy"/>
Access Token	<input type="text" value="3736207823-obljtbwN90wPdrXnajZfQrjp76wG7iESiwqlsnF"/>
Access Token Secret	<input type="text" value="8sElbaC7aa5rBp1Ti3Xi1IoRpu1zCdMO5Jw1LwEBcOszs"/>

Proxy Settings

HTTP Proxy	<input type="text" value="cache-etu.univ-lille1.fr"/>
Port	<input type="text" value="3128"/>

Load custom dictionaries

Locate a dictionary	<input type="button" value="Choose a file"/>
Choose the type	<input type="text" value="Positive Dictionary"/>
Import your dictionary	<input type="button" value="Import"/>

Figure 8: L'écran de paramétrages de l'application.

5.5 L'onglet analyse

Il permet de lancer les tests par Cross-Validation en K-Fold. Vous pourrez configurer le nombre de subdivisions en le spécifiant dans le champ en haut.

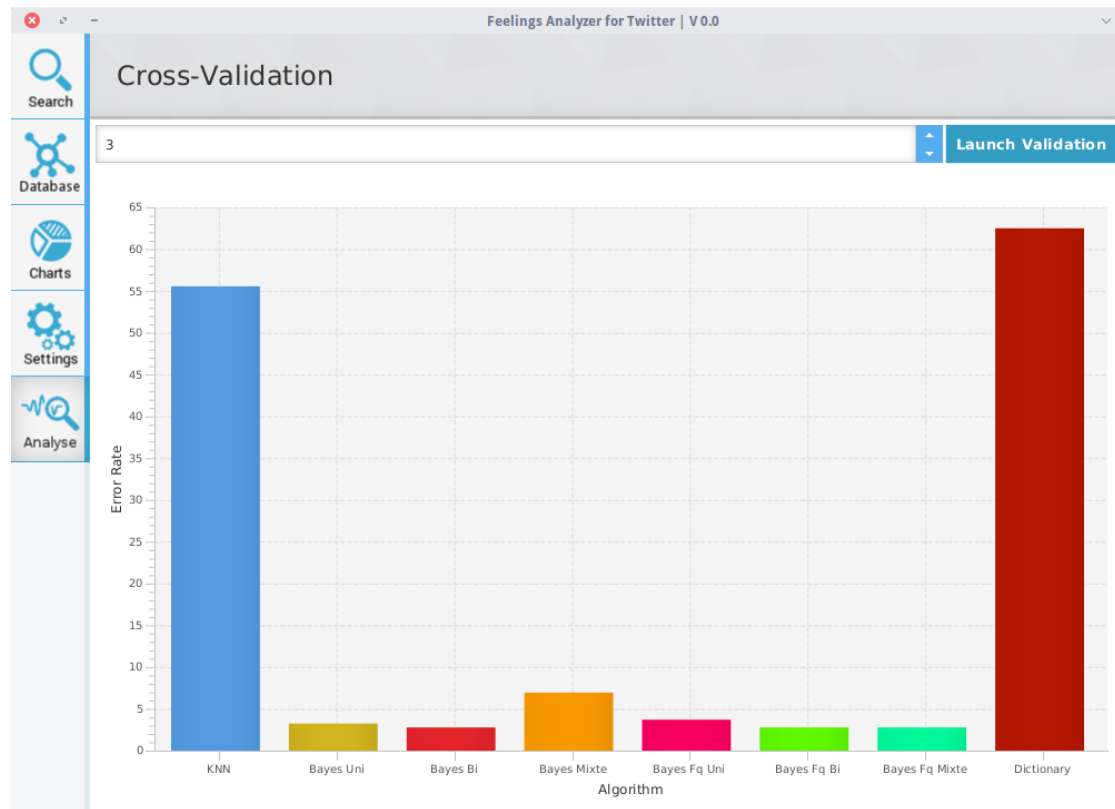


Figure 9: L'écran d'analyse par Cross-Validation en K-Fold.

6 Conclusion

6.1 Objectifs

Voici un résumé des objectifs accomplis :

1. Mise en place d'une architecture logicielle en couche, reprenant les patrons de conception : MVC et DAO (avec SqliteImpl).
2. Récupération des tweets à l'aide de Twitter4Java et nettoyage de ces derniers à l'aide de Regex.
3. Implémentation des algorithmes : Mot-clès, KNN, Bayes n-gramme, BayesFrequency n-gramme.
4. Développement de fonctionnalités secondaire tels que : la gestion des settings, base de données, chart, ainsi qu'une implémentation du système d'analyse expérimentale.
5. Conception d'un look and feel respectant la charte graphique de Twitter à travers JavaFX.
6. Réflexions ergonomiques.

6.2 Futures améliorations

Voici une liste d'idée d'améliorations à apporter à l'application :

1. Permettre l'ajout d'algorithme à la façon d'un plugin.
2. Améliorer la gestion de la base de données.
3. Démutualiser l'application, avec une version client-serveur dans le but d'améliorer les temps de calcul des algorithmes.