Rapport Machine Learning – Reconnaissance des émotions dans un texte

Table des matières

Introduction 1
Données d'apprentissage 2
Déroulement de l'application 2
Nettoyage des donnée 2
Recherche des données adéquates 2
Model d'apprentissage et résultat 3
Apprentissage continu 3
Manuel 3
Automatique 3
Analyse et résultat 3
Axe d'amélioration 4
Conclusion 4

Introduction

L'application aura pour but d'identifier les émotions que transcrit une phrase ou un texte. Pour cela, le machine learning sera utilisé pour pouvoir prédire les émotions. Les émotions qui seront identifiées sont listées ci-dessous :

- Tranquillité
- Surprise
- Joie
- Tristesse
- Dégoût
- Colère
- Fureur
- Peur
- Terreur
- Coupure avec ses émotions

Un apprentissage automatique et manuel est aussi introduit dans l'application pour un apprentissage continu.

Données d'apprentissage

L'application utilise deux tables de données :

- Une table référençant par un id les différentes émotions
- Une table avec les mots et expression lié par un id d'émotion

Ces données ont été récupérées sur internet, dans plusieurs sites. Des listes de mots ou d'expressions liées par des émotions y sont référencées.

Déroulement de l'application

L'application intègre un menu qui permet :

- De prédire une émotion pour un texte.
- D'ajouter un mot ou une expression liée à une émotion dans la table d'apprentissage.
- De quitter l'application.

Nettoyage des données

La première étape de l'application est le nettoyage de donnée. C'est lui qui permet en grosse partie l'optimisation d'une application de prédiction.

Dans un premier temps, le texte est découpé en différents mots qui lui composent dans un tableau. Par la suite, nous enlevons chaque ponctuation et chaque « stopword » (pronoms, mot de liaison, etc…) qui n'ont pas de sens à proprement parler dans un texte ou phrase. Ensuite, nous appliquons un « stemming » sur chaque mot du tableau. Il permet d'avoir seulement les racines des mots (il permet un meilleur moyen de rechercher les données adéquates que j'expliquerais par la suite)

Nous utilisons la librairie NLTK qui regroupe tout un lot de fonction qui permet l'analyse de texte et ainsi le nettoyage de donnée.

Recherche des données adéquates

La recherche de données se fait en deux temps : par les mots puis par expressions.

Pour choisir les données adéquates, le logiciel regarde dans la base de données des mots et expressions d'apprentissage et il essaye de trouver une correspondance dans le texte à traiter. Pour cela, le stemming est important. En l'appliquant, sur chaque mot du texte et pour chaque mot de la BDD. Je peux faire une comparaison en trouvant les mots de la même famille.

Pour les expressions, je fais la même chose sauf qu'à la fin du nettoyage de données, je remets en texte la table des mots nettoyés et j'applique une fonction « find() » qui me permet de retrouver dans un texte une expression.

Par la suite, je stock dans une table tous les mots ou expressions que j'ai trouvé. Ils seront par la suite traités dans un modèle d'apprentissage pour prédire une émotion.

Model d'apprentissage et résultat

Pour le model d'apprentissage, l'application utilise « nltk.NaiveBayesClassifier » de la librairie NLTK. C'est un modèle de classification adapté pour les mots. Pour sa création, je lui mets en entrée ma table d'apprentissage. Il va par la suite créer un modèle où il aura classifié chaque mot et expression par émotion.

Pour chaque mot et expression que l'application a trouvé dans le texte, il va lancer une prédiction. J'aurais alors une table avec la liste des émotions pour chaque mot et expression trouvé.

S'il y a plusieurs émotions différentes, le programme va choisir l'émotion avec la plus grande occurrence. Nous avons donc à la fin une émotion prédite pour le texte à analyser.

Apprentissage continu

L'application comporte deux sortes d'apprentissage continu listé ci-dessous :

Manuel

Dans le menu, en tapant "2" nous arrivons dans la partie apprentissage manuel. L'utilisateur pourra ajouter dans la table d'apprentissage un mot ou une expression qu'il liera avec l'émotion qu'il choisira.

Automatique

L'apprentissage automatique s'appliquera à chaque test de prédiction. Pour cela, l'application mettra dans la table d'apprentissage chaque phrase du texte qui est analysé. Chaque phrase sera liée avec l'émotion prédite du test de prédiction.

Analyse et résultat

Pour les tests de score de réussite, je ne pouvais pas le faire sur des phrases étant donné que mes données d'apprentissage et de test sont des mots et expressions. Je n'ai pu faire qu'avec des mots et expressions.

Le score révélé par ces tests est de 0.814. C'est un score honorable pour un bon fonctionnement d'une application prédictive.

Comme la base de données n'avait seulement que deux types de données (expression et émotion), peu d'analyse a été faite. La principale analyse a été faite lors du nettoyage de données pour savoir comment identifier clairement les bonnes données à traiter. Celui-ci s'est tourné vers les mêmes familles de mots en utilisant le stemming.

Ce que j'ai remarqué par mes tests sur des phrases entières et des textes, c'est une meilleure prédiction pour les phrases simples. Pour les textes de plusieurs phrases, les résultats peuvent être erronés. J'explique cela par le fait :

- qu'un texte peut avoir plusieurs émotions
- la table d'apprentissage n'est pas assez complète

- plus un texte sera long, plus il sera complexe à l'analyser.

Axes d'amélioration

L'application est loin d'être optimisée. Plusieurs axes d'améliorations sont à prévoir pour rendre le logiciel plus performant. Voici différents points à améliorer:

- Avoir une gestion des négations. Si une négation existe dans une phrase, on pourrait prendre le sens contraire de l'émotion prédis. Exemple : « je ne suis pas en colère », l'application prédirait la « colère » mais avec la négation « ne pas », il le changerait en « joie » ou en neutre.
- Mettre en place un système de poids selon les expressions ou les mots et cela pèserais sur l'émotion choisit. Par exemple, une expression aurait normalement un poids supérieur par rapport à un mot car une expression à la plupart du temps une seule signification, au contraire un mot peut avoir plusieurs signification.

Conclusion

En conclusion, l'application est performante pour des phrases simples et moins pour des textes assez longs. L'algorithme n'est pas parfait et il reste des axes d'améliorations. Sur internet, on peut trouver beaucoup d'analyse sur les sentiments d'un texte mais seulement d'un point vue positif et négatif. Un groupe de projet avait fait une analyse par rapport à différente émotion comme mon projet, mais malheureusement je n'ai pas pu retrouver le lien vers celle-ci.