Rapport Classification PAN4

Dans ce module de Classification, le binôme a été chargé de relié des textes mis sous format .txt avec des ambiances définies par le groupe PACT. Le signal de sortie est une chaîne de caractères correspondance à l'ambiance associée. Pour la réalisation de ce projet, le binôme a dû créer une fonction permettant de traduire les textes pour pouvoir les traiter vectoriellement, constituer une base d'entraînement pour des tests de machine learning et enfin trouver, organiser et optimiser des algorithmes de machine learning. Le travail a été effectué dans le langage Python.

Elèves : BECART Thierry
NAIT SAADA Thiziri

Expert : MURENA Pierre-Alexandre

I – Vectorisation des textes

Avant d'essayer de ranger les différents textes dans des classes d'ambiances, nous avons besoin de normaliser les données des textes sous une forme facilement interprétable pour un compilateur. Les fichiers à classer sont des fichiers textes .txt. Nous les traduisons en vecteurs où chaque composante sera le poids d'un certain mot.

Dans un premier temps le vecteur aura pour taille la taille de notre vocabulaire reconnaissable. Ce vocabulaire reconnaissable est l'ensemble des mots pouvant une information sur l'ambiance du texte. Pour reconnaître ces mots, nous avons besoin de les réduire en leur radical (e.g. : was = were = be) Puis de supprimer les mots vides de sens (verbes d'état, les déterminants, les adverbes d'introduction circonstancielle...). Ces deux étapes sont respectivement la lemmatisation et l'élimination des « stop Words ». Ces deux étapes utilisent les méthodes des modules nltk et sklearn.

Une fois le texte réduit en des mots facilement reconnaissable et ayant tous un impact sur l'ambiance du texte, on peut traduire le texte en un vecteur. Pour cela, nous utilisons la distance TF-IDF qui va donner dans un corpus de texte l'ensemble des vecteurs associés. Le poids des mots dans chaque vecteur va croître avec la fréquence dans le texte du vecteur mais va aussi décroître selon la fréquence d'apparition dans l'ensemble des textes du corpus. En effet si un mot est présent partout, il ne permet plus de distinguer une ambiance d'une autre. La méthode a été recodée mais nous utiliserons la version optimisée de la bibliothèque sklearn.

Puisque le vocabulaire contient une grande quantité de mots différents (plus de 200 000), nous devons alléger les calculs en réduisant la taille des données enregistrées. Pour cela nous allons utiliser une méthode de compression de données PCA qui va chercher les composantes principales des vecteurs et ainsi les projeter sur une dimension inférieur. Nous avons ainsi réduit la dimension de plus de 50 000 à 30.

II – Élaboration d'une base de données d'entraînement

La classification des textes devenus vecteurs se fera via machine learning. Il est donc indispensable de disposer d'une base de données d'entraînement.

Cette base de données sera un dossier DataSet qui sera lui même décomposé en sous-dossiers de titre l'ambiance des textes qu'ils contiennent. Nous prenons les textes à partir d'un site libre de droits : *descriptionari.com*. En recherchant un mot dans la barre de recherche, le site nous renvoie sur des mots rattachés au mot recherché. En cliquant sur ces mots reliés on arrive sur une

page avec beaucoup de textes libres de droits. On choisi donc les bons sous-mots clés et on récupère l'ensemble des textes.

Pour récupérer l'ensemble des textes nous avons créé un programme qui d'après le code source de la page récupère chaque texte de la page et les enregistre en des fichiers textes sous le nom « AmbianceN.txt ».

Nous avons ainsi récupéré environ 5000 textes répartis dans 44 ambiances.

III – Choix d'un algorithme de classification

Les textes étant maintenant vectorisés, il faut dorénavant les classer dans les ambiances. Pour cela nous avons fait appel à des algorithmes de machine learning de la bibliothèque sklearn. Nos 4 candidats principaux sont : Naive Bayes, KNN, Decision Tree et Linear SVC.

Pour juger de leur efficacité, nous avons créé un arbre des ambiances permettant de déterminer lorsque la prédiction d'une ambiance est mauvaise si on a un résultat acceptable ou non. On obtient ainsi un premier meilleur résultat à 41 % de réussite ce qui est certes pas au-delà de la moitié des résultats justes mais qui est déjà 18 fois plus précis que le hasard.

Par rapport au choix du classificateur, on a écarté Decission Tree parce qu'il est vraiment peu efficace dans notre situation. Decision Tree fonctionne surtout lorsque les données de test sont très proches des données de d'entraînement. Naive Bayes se repose sur l'utilisation de probabilité en estimant la probabilité que le texte a d'être dans chaque ambiance. On n'obtient qu'un résultat aux environs des 20 %. Les méthodes KNN et Linear SVCont toutes les 2 de très bons résultats, mais ceux de Linear SVC sont meilleurs de 5 points.

On a choisi le meilleur classificateur, on a alors essayé d'optimiser les résultats. On a repris l'arbre de l'ensemble des ambiances et on l'a transformé afin d'en faire un arbre de décision. Chaque nœud représente un classifier binaire entre les enfants de droite et ceux de gauche. Les ambiances de sortie sont les feuilles de cet arbre. Chaque classificateur suit une méthode de linear SVC dont les paramètres ont été optimisés.

Conclusion :

Le module de classification a pu remplir les différentes tâches que le groupe lui a confié avec des résultats assez satisfaisants. Pour une meilleure classification, nous devrions élargir la base de données d'entraînement à l'échelle de plusieurs millions de textes et peut-être aussi réduire le nombre d'ambiances possibles en sortie. Aussi, il faudrait pouvoir récupérer les résultats du classificateur final qui demande beaucoup de ressources pour être évalué.