TPSA : Conception et implémentation d'un algorithme d'analyse de sentiment basé sur les aspects

Démarches et choix

Objectif 1 :

Dans le but de calculer la polarité des mots contenus dans les phrases des jeux de données. J'ai commencé par faire un pré-traitement incluant la tokenisation, le PoS tagging et la reconnaissance d'entités nommées (NER) dans le code python intitulé « pre-processing.py ». Ce code m'a permis de traiter les deux fichiers train et test gold pour avoir finalement 4 fichiers prétraités (Laptop_Train_Pre_Processed.txt, Laptop_Test_Gold_Pre_Processed.txt, Restaurants_Train_Pre_Processed.txt, Restaurants_Test_Gold_Pre_Processed.txt).

Voici l'extrait d'un fichier (Laptop_Test_Gold_Pre_Processed.txt) :

> POS Tags: [('I', 'PRP'), ('liked', 'VBD'), ('the', 'DT'), ('aluminum', 'NN'), ('body', 'NN'), ('.', '.')]
>
> Named Entities: (S I/PRP liked/VBD the/DT aluminum/NN body/NN ./.)

Le script utilise NLTK pour la segmentation en mots, l'attribution des catégories grammaticales, et l'identification des entités. Les données textuelles sont ainsi décomposées en tokens, chaque mot se voyant attribuer une étiquette grammaticale (tag PoS) correspondant à sa fonction dans la phrase. Par exemple, les tags 'NN' pour les noms communs ou 'VBD' pour les verbes au passé, permettent ensuite de contextualiser les termes d'aspect en fonction de leur rôle syntaxique. Les entités nommées sont également extraites et classées, comme dans l'exemple suivant où 'aluminum body' est identifié comme une séquence de noms, signalant un aspect potentiel du produit.

J'ai ensuite extrait les termes d'aspects et leur polarité à l'aide de SentiWordNet, toujours sur les 4 fichiers précédents et dans le code python intitulé « polarity-identifier.py ».

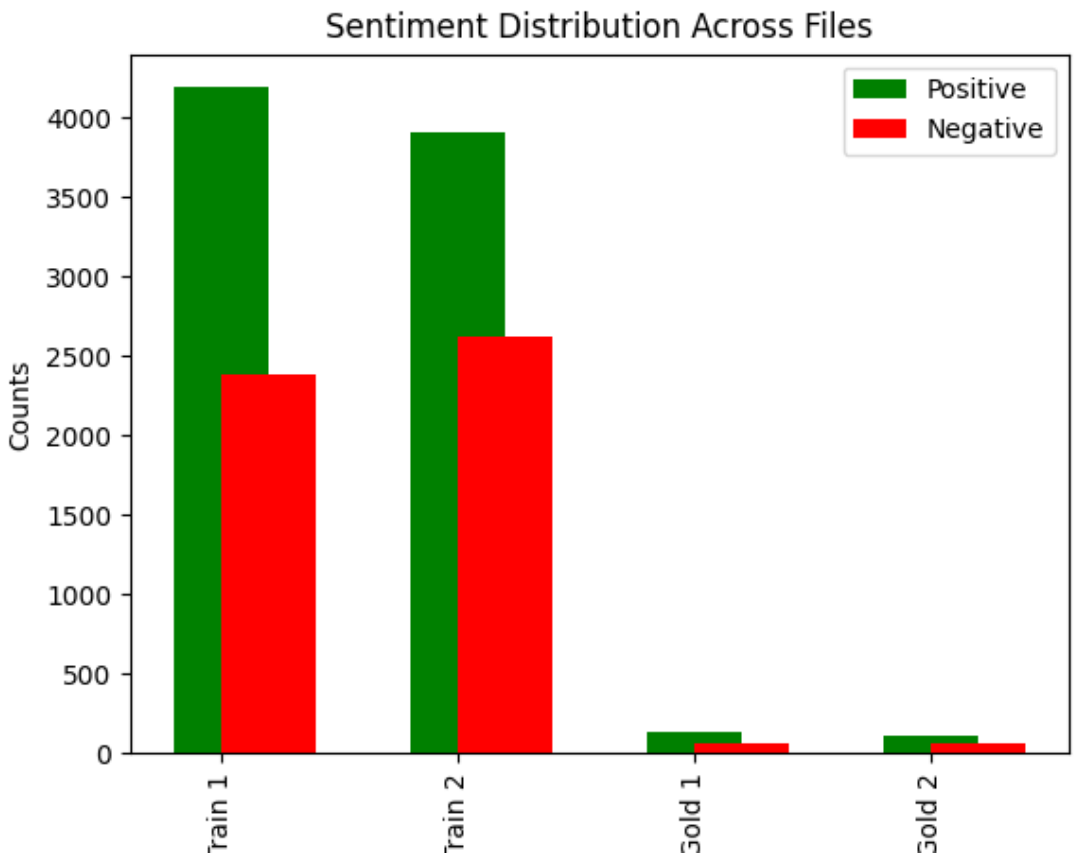
Voici l'extrait d'un fichier (Laptop_Test_Gold_Analysis_Processed.txt) :

> ('liked', 'VBD', 0.125, 0.0) ('aluminum', 'NN', 0.0, 0.0) ('body', 'NN', 0.0, 0.0) ('Lightweight', 'NNP', 0.0, 0.0) ('screen', 'NN', 0.0, 0.0) ('beautiful', 'JJ', 0.75, 0.0) ('Buy', 'VB', 0.0, 0.0) ('love', 'VB', 0.5, 0.0) ('promise', 'VBP', 0.0, 0.0) ('regret', 'VB', 0.0, 0.5) ('quality', 'NN', 0.375, 0.0) ('performance', 'NN', 0.125, 0.0) ('expected', 'VBN', 0.25, 0.25) ('Apple', 'NNP', 0.0, 0.0)

Le script extrait la polarité des mots à l'aide de SentiWordNet, en associant à chaque terme une note positive ou négative. Chaque mot, à l'exception des mots fonctionnels sans contenu émotionnel, est évalué pour son impact sentiment. La sortie fournit un ensemble de données où chaque terme pertinent est accompagné de ses scores de sentiment, reflétant l'orientation positive ou négative exprimée dans le texte.

Valentin Charles – M2 IA2

Et pour finir dans l'Objectif 1, j'ai visualisé les données de distribution des sentiments des 4 fichiers à l'aide de matplotlib dans le fichier python intitulé « data_visualization.py », m'ayant donné la sortie suivante :



Restaurant_Train étant le train 1 et Laptop_Train étant le test 2, Restaurant_Test_Gold étant le Gold 1 et Laptop_Test_Gold étant le Gold 2.

Objectif 2 :

Le script aspectTermsPolarity-RuleBasedApproach.py intègre les trois premiers points de l'Objectif 2, focalisant sur les mots entourant l'aspect pour déduire la polarité. La logique est basée sur la proximité syntaxique et les scores de SentiWordNet.

Mon approche à règles effectue une analyse de sentiment par fenêtre contextuelle autour des termes d'aspect, en tenant compte des scores de polarité issus de SentiWordNet pour chaque mot. La pondération de ces scores est ajustée selon la catégorie grammaticale en accordant une importance accrue aux adjectifs et adverbes. Cette méthode permet d'attribuer une polarité globale à chaque terme d'aspect en sommant et normalisant les scores pondérés dans un voisinage défini.

Ayant d'abord implémenté une approche à règle dans le premier script, j'ai obtenu une précision de 53.1%. N'étant pas satisfait par celle-ci donc j'ai également d'implémenter une version avec le l'apprentissage automatique dans le script « aspectTermsPolarity-MachineLearningApproach.py ».

Ce dernier script reprend les mêmes pré-traitements mais applique un RandomForestClassifier sur des vecteurs TF-IDF, ce qui a mené à une amélioration significative avec une précision de 70%.

En conclusion, mon travail réalisé a démontré l'importance de la préparation des données et du choix de la méthode d'analyse. L'approche d'apprentissage automatique, bien qu'étant plus complexe à mettre en œuvre, m'a offert une meilleure performance par rapport à l'approche basée sur les règles.