Cahier d'expérience

Séance n°1: 3 II 2023

Recap:

- L'enjeu est de distinguer les revues positives et négatives et d'aller plus loin et de dire ce qui est positif autour d'un mot en particulier. Avoir un tagging au niveau des aspects pour savoir ce que les gens aiment ou n'aiment pas.
- On a plusieurs jeux de données:
    - site de référence des jeux de société 'trictrac' qui référençait plus de mille jeux de société
      ~200 000 avis
      scrap de ce site en mongoDB
        - description des jeux
        - avis des jeux (utilisateurs, review, note sur 10)
    - jeu de données twitter sur les élections présidentielles 2017 (en français)
    - jeu de sous-titres de séries (en anglais)
- travail à faire sur les données : BDD à mettre sous forme relationelle
- TAL pour faire de la classification de sentiments. Par rapport aux notes d'abord car on n'a pas de label. On pourrait aussi éventuellement avoir accès aux informations du forum (cf scrap Olivier)
- recommandation dépend du profil, IP. Algorithme qui analyse les traces, visites, les co-occurences : filtrage collaboratif.
    - On regarde les personnes qui ont commenté positivement aux différents jeux. Possibilité de regrouper en clusters tous les jeux qui sont commentés positivement (ie. négatifs)
    - recommandation par rapport aux notes ou par rapport à la revue (plus d'aspects sémantiques → TAL ++, comprendre ce que les gens aiment/ n'aiment pas dans ce jeu)
    - Algo d'apprentissage de représentation
    - Analyse du contenu (description du jeu). Metadonnées propres à chaque jeu: qu'est-ce qui est en commun?
- ouverture : point de vue ergonomique : comment on présentante à l'utilisateur, GNN

Rapport (20pages) Soutenance (10min)
Créer git, GoogleDoc, Mattermost

Travail perso effectué:

On a chargé les données sur la plateforme MongoDB téléchargée au préalable.
On a analysé les données et leur sémantique et on a réalisé des statistiques dessus :

- histogramme de la répartition des notes
    - *utiliser la médiane pour l'obtention de classes équilibrées? Mais médiane=8 donc qu'en est-il de l'interprétation selon laquelle un 7 est une mauvaise note ?*
- calcul de la moyenne, médiane, écart type
- à l'aide des techniques de preprocessing on a commencé à traiter les avis relatifs aux jeux
    - Word Cloud avec stop words

- Word Cloud sans stop words : *on a remarqué que les avis étaient biaisés par le vocabulaire (bcp d'apparition du mot 'jeu') par exemple donc peut être créer une nouvelle liste de stopwords?*
- Odds répartition nous donne les mots les moins fréquents NB: voc noms propres et english words.

Loi de Zippf sur les mots: droite sur une courbe en loglog. Même chose sur les postes, la grande majorité des utilisateurs ont seulement 1 ou 2 postes. Ce qui nous intéresse est de connaître les utilisateurs qui ont posté beaucoup d'avis pour comprendre pourquoi ça fait sens d'aimer (ou pas) les mêmes jeux.

Report des données statistiques

Sémantique des jeux

Séance n°2: 10 II 2023

Recap

Carnet de bord: présenter le projet en 1 ou 2 pages.

Dans un rapport, il faut présenter ce qui a été fait avant. Présenter les méthodes qu'on va utiliser.

Chaque groupe a discuté de ce qui avait été fait durant la semaine

- Prix pour la recommandation

Groupe MAG

- retirer l'identifiant généré lors du scrapping
- retirer duplicatas
- traiter le casting
- 184 catégories différentes : essayer de les 'clusterer', réduire nbr de catégories

Travail perso effectué:

- Pipeline de prétraitements TAL
- vectorizer pour les mots les plus fréquents, choisir un seuil
- Statistiques (répartition des notes, médiane, moyenne, variance, écart type, jeux avec le plus d'avis, les mots les plus fréquents, bigrammes et trigrammes pour, stopwords, odds ratio les mots les plus discriminants entre les deux classes basées sur la moyenne à ce moment
- Mise en forme de la BD avec mySQL

Remarques:

- Aggréger user?
- Recommandation → Algo analyse trace visite, filtrage collaboratif, algo apprentissage représentation, clustering
- Intégration CNN?

Séance n°3: 17 III 2023

Recap

Consolider chaîne de prétraitement

Savoir combien de fois un utilisateur a posté, quelle est la note moyenne d'un utilisateur?

Tracer histogramme des moyennes des notes pour voir quels sont les différents profils d'utilisateurs, écart type pour avoir une visualisation de qui sont 'méchants' dans leurs commentaires...

Travail effectué

- On a fait un classifieur qui prend description et essaye de prédire la catégorie. Classifieur linéaire : poids porte sur chaque dimension, prend en entrée la description/avis et prédit la catégorie/note

Remarque : Quelles sont les features qui ont le plus de poids en valeur absolue?

- On a continué à nettoyer données
- Prédire la note en régression ou classification avec la moyenne c'est une méthode trop naïve, plutôt utiliser médiane pour avoir des classes équilibrées. Question d'interprétation, on ne sait pas comment le système est biaisé...
- Faire un lissage de la coupe des notes (utiliser fenêtre parzen pour approcher la distribution des notes) parce qu'avec l'histogramme on ne voit pas le détail des fluctuations.

Séance n°4: 24 III 2023

Recap

Groupe MAG

- Clustering des catégories, les jeux ont beaucoup de catégories, tf idf sur les tags des catégories (ensemble de mots) mais count vectorizer suffit
- Regrouper catégories syntaxiquement avec K-means
- 150 clusters, bcp trop donc réduit encore la taille en agrégeant cluster en catégorie.
- Prennent les catégories qui apparaissent plus de 6 fois dans la BD

Remarque : ne pas faire l'hypothèse que chaque jeu n'aura qu'une catégorie → classification multiclasse : faire 1 contre tous et faire un ranking derrière en sélectionnant les 3/4 premiers...

- bcp de jeux n'ont pas d'avis donc trictrac met automatiquement 0 donc supprimer quand il y a aucune description.
- Entraîner classifieurs avec un test split avec argument stratify y pour prendre en compte l'équilibre des classes), classification des prédiction des catégories
- 3 algo (NB, SVM (jouer avec pénalisation) et randomforest (classifie directement en catégorie, jouer avec nbr d'arbres)) f1 ~ 53%

Travail effectué

- Estimation de la densité, prétraitement, comptage des occurrences des mots et loi de zipf, dans les mots les moins fréquents c'est svt les erreurs de frappes
- Rééquilibrage des classes (on considère 11 classes et on veut avoir le même nbr de texte par classe) pour découper les notes en intervalles équivalents et entraîner les classifieurs : but est de faire une Stratification

Remarque: intérêt à faire une fonction qui prend en entrée une liste de labels, et qui retourne les index sélectionnés (qui correspond au nbr minimum d'exemples). Rendre le code le plus générique possible
Méthode plus simple: pour chaque label, récupère uniquement les index correspondant et en tirer x au hasard. Faire une seule boucle for sur les labels et retourner les appartenances au label
Utiliser Fonction random choice plutôt que faire accumulateur, puis faire merge de liste et shuffle
Quand on travaille de manière dynamique plus facile de travailler sur des listes qu'on peut append, pas besoin de la lourdeur de numpy

But du rééquilibrage pour avoir un classifieur non biaisé, mais on ne veut pas appauvrir la connaissance
Pas de phénomène d'itération sur le gradient comme avec perceptron par ex

On veut trouver la meilleure combi de pré processing et la garder car ça coûte cher.

- Prédiction de notes à partir des commentaires

Remarque : faire un vectorizer et fit transform au début, pour traiter une bonne fois pour toute les données, si on teste un nv paramétrage à chaque fois on refait un new vectorizer

- Joue sur les données (taille dic, bigrammes,..) plutôt que sur les modèles

Remarque: *Prétraiter toutes les données et enregistrer différents corpus avec différents paramétrages (bigramme, trigramme ...) dans pickle. Découper en train/test, dans train rééquilibrer les classes mais pas dans le test. J'apprends mon modèle, je teste sur test. Puis sur test rééquilibré pour voir comment s'est biaisé*
Stocker résultats intermédiaires

Certaines perf sont liées au pré-processing et certaines sont liées au modèle qu'on choisit, on veut tout croiser. Evaluation modèle, comparer les différents modèles, trouver le meilleur hyperparamètre pour un modèle donné en fonction du préprocessing qu'on va donner. Utiliser test split ou cross validation

- On utilise les algo NB, LR, SVM et tester random forest en plus 45% d'accuracy (test aussi équilibré)

Tester les deux, il faut un test déséquilibré pour représenter la réalité avec une vraie répartition. Cross fold (pour voir tous les exemples) avec vraie dstn et moment de fit qu'on rééquilibre uniquement

- on a fait avec les mots les plus fréquents, faire aussi avec les mots les moins fréquents

NB mieux que SVM avec trigramm ? bizarre revoir pénalisation de SVM

- Prédiction avec séparation des notes en 2 classes : Très bien

Cacher les preprocess dans fichier python qu'on utilisera comme module et l'importer uniquement pour tracer résultats. Prendre vrai IDE, coder fonctions dedans et dans notebook utiliser uniquement les fonctions (qui font fold cross, qui rendra meilleur estimateur, meilleur classifieur et interpréter ces res ds notebook)