PPII «PROJET PLURIDISCIPLINAIRE D'INFORMATIQUE INTÉGRATIVE» (2019-2020)

ANALYSE DES SENTIMENTS À PARTIR DE CRITIQUES DE FILMS

Sébastien Da Silva <**sebastien.dasilva@telecomnancy.eu**>
Olivier Festor <**olivier.festor@telecomnancy.eu**>
Gérald Oster <**gerald.oster@telecomnancy.eu**>

> *Ce projet est une adaptation libre du devoir (nifty assignments) «Movie Review Sentiment Analysis», Eric D. Manley and Timothy M. Urness, Drake University (2016). http://nifty.stanford.edu/2016/manley-urness-movie-review-sentiment/*

PRÉLIMINAIRES

Objectifs

Ce projet a pour but de vous confronter à la résolution d'un problème réel en mobilisant les connaissances et compétences acquises sur les modules de l'UE STIC et Gestion de Projets de première année à TELECOM Nancy. Il vous permet d'acquérir des compétences en programmation modulaire et de développer votre expérience dans l'usage du langage C. La mise en œuvre des connaissances acquises en structures de données (choix, implantation, évaluation, utilisation) vous est également demandé, ceci afin de modéliser et résoudre efficacement un problème sur des données conséquentes. Pour cela, il vous est imposé de :

- Concevoir/implémenter des structures de données adaptées à un traitement informatique particulier,
- Réaliser plusieurs composants logiciels informatiques dans le langage C,
- Réaliser des entrées-sorties (écriture/lecture fichiers, lecture/saisie terminal, etc.),
- Évaluer la qualité des solutions proposées et des résultats obtenus,
- Mettre en œuvre des bonnes pratiques en génie logiciel et gestion de projets.

Modalités

Ce projet est à réaliser en groupe (la constitution du groupe étant imposée).

Vos livrables seront composés :

- des codes sources de vos développements logiciels ;
- des documentations nécessaires à la compréhension (conception, notes de développement), l'installation, la compilation, l'exécution, la validation de vos réalisations ;
- de **tous les éléments de gestion de projet** que vous aurez produits (compte-rendus de réunion, planification et répartition des tâches, analyse post-mortem des efforts individuels et de l'atteinte des objectifs, etc.) **Au vue de la situation actuelle, nous nous attendons à ce que cette partie soit plus détaillée que d'habitude**. Les descriptions des méthodes d'organisation et de coopération ainsi qu'un retour d'expérience sous la forme d'un rapport d'étonnement d'environ 1/2 page par contributeur seront particulièrement étudiées par les évaluateurs.

Tous vos écrits pour le PPII seront rédigés, synthétiques mais complets (nombre de pages libre – soyez raisonnable – mais impérativement au format pdf rédigé en LaTeX), et les fichiers sources (`.tex` , figures, etc.) seront également à déposer dans votre dépôt git.

L'ensemble de ces livrables seront déposés sur le dépôt git qui est dédié à votre groupe de travail (sous-projet du projet https://gitlab.telecomnancy.univ-lorraine.fr/ppii-2k20)

Nota Bene. Ne trichez pas ! Ne copiez pas ! Ne plagiez pas ! Si vous le faites, vous serez lourdement sanctionnés. Nous ne ferons pas de distinction entre copieur et copié. Vous n'avez pas de (bonnes) raisons de copier. De même, vous ne devez pas utiliser un produit clé en main trouvé sur internet.

Dates de rendu

Une première version aboutie de votre projet devra être disponible sur votre dépôt git pour le **lundi 25 mai 2020 à 23 heures.** Cette version devrait fournir la version a minima de votre réalisation logicielle.

La version finale est attendue pour le **dimanche 7 juin 2020 à 23 heures**. Elle intégrera le rapport relu (incluant les attestations de non-plagiat signées) et tous les documents complémentaires et annexes produits.

Point sur la notation

Tout d'abord, **pour chaque groupe de travail, le non-respect des consignes ici présentes pourra entraîner jusqu'à 5 points de pénalité pour tous ses membres**.

La documentation et les tests que vous fournirez compteront tout autant que le code que vous produirez. Vous n'oublierez pas de citer toutes vos sources. Et **évidement comme il s'agit d'un projet, vous savez pertinemment que sa gestion sera évaluée**.

Lors de la rédaction de votre rapport et du développement de vos algorithmes, vous vous emploierez à réfléchir aux étapes avant de les mettre en place. Il sera primordial que chaque membre du groupe puisse interpréter/expliquer chaque ligne du code. Dans le cas contraire, la note du trinôme entier sera dépréciée.

Le travail doit être réparti équitablement entre les membres d'un groupe et **tout membre doit contribuer à toutes les activités de celui-ci** (spécification, gestion de projet, développement logiciel, test, ...). Le rôle et la part de chacun doit être clairement identifiée et documentée dans les rendus.

THÉMATIQUE DU PROJET : ANALYSE DES SENTIMENTS À PARTIR DE CRITIQUES DE FILMS

L'analyse des sentiments est un problème de fouille de données sur de grandes quantités de données (Big Data) où l'on cherche à déterminer l'attitude générale d'un écrivain à partir de contenus qu'il a rédigé. Dans ce projet, nous souhaitons que vous réalisiez un programme qui soit en mesure d'analyser des commentaires (critiques) de films et décider qu'un nouveau commentaire tel que « Ce film était une bouffée d'air » (*« The film was a breath of fresh air »*) est positif alors que « Cela m'a donné envie d'arracher mes yeux » (*« It made me want to poke out my eye balls »*) est lui négatif.

Un algorithme basique qu'il est possible d'utiliser pour réaliser cela consiste à:

- associer une valeur numérique à chaque mot. Cette valeur numérique représente le poids positif ou négatif de ce mot dans un commentaire. Pour calculer simplement ce poids, il est possible de calculer la moyenne des scores des commentaires où ce mot apparaît.

- calculer le score d'un nouveau commentaire en calculant la moyenne des valeurs numériques associées aux mots constituant ce commentaire.

C'est cet algorithme que vous allez mettre en œeuvre dans la première partie de ce projet. Vous allez être amenés à parcourir un fichier contenant un ensemble de critiques de films (en langue anglaise) publiées sur le site web **Rotten Tomatoes** (https://www.rottentomatoes.com/) qui permet à ces utilisateurs de rédiger à la fois un commentaire sur un film et d'y associer un score numérique. Vous allez utiliser ces données pour *apprendre* quels sont les mots qui contribuent positivement ou négativement dans une critique.

Données fournies

Le fichier de données fourni contient un extrait de 8529 critiques en respectant la structure suivante :

```
[...]
3 As tricky and satisfying as any of David Mamet 's airless cinematic
shell games .
1 I 'm sure the filmmaker would disagree , but , honestly , I do n't see
the point .
4 Jones has tackled a meaty subject and drawn engaging characters while
peppering the pages with memorable zingers .
4 Bloody Sunday has the grace to call for prevention rather than to place
blame , making it one of the best war movies ever made .
1 Takes a clunky TV-movie approach to detailing a chapter in the life of
the celebrated Irish playwright , poet and drinker .
2 Finally coming down off of Miramax 's deep shelves after a couple of
aborted attempts , Waking Up in Reno makes a strong case for letting
sleeping dogs lie .
0 Thanks largely to Williams , all the interesting developments are
processed in 60 minutes -- the rest is just an overexposed waste of film .
[...]
```

*Extrait du fichier **data/movie_reviews.txt***

Vous pouvez constater que chaque critique est écrite sur une ligne. Chaque ligne commence par un entier (de 0 à 4) correspondant à la note associée à la critique. Cette note correspond à l'avis suivant :

- 0 : négatif
- 1 : quelque peu négatif
- 2 : neutre
- 3 : quelque peu positif
- 4 : positif

Vous pouvez également remarquer que les commentaires de ce fichier ont été prétraités : les signes de ponctuation ont été détachés des mots (en les séparant par des espaces), certains mots spécifiques ont été découpés (par exemple, « don't » a été découpé en « do » et « n't »).

Attendus

Votre programme devra assurer a minima les fonctionnalités suivantes :

1. chargement d'un fichier texte qui contient les commentaires (critiques) de films ainsi que leurs score (note de 0 à 4) ;

2. calcul du poids pour chaque mot présent dans ces commentaires ;
3. demande de saisie par l'utilisateur sur l'entrée standard du terminal d'un nouveau commentaire ;
4. calcul du score prédit pour ce nouveau commentaire ;
5. affichage du score et du qualificatif (a minima « négatif », « neutre » ou « positif »).

Exemples d'exécution du programme

```
~/ppii-2k20 $ ./analyse ./data/movie_reviews.txt
enter a review -- Press return to exit:
A weak script that ends with a quick and boring finale
The review has an average value of 1.79128
Negative Sentiment

~/ppii-2k20 $ ./analyse ./data/movie_reviews.txt
enter a review -- Press return to exit:
Loved every minute of it
The review has an average value of 2.39219
Positive Sentiment
```

En termes de structure de données votre algorithme reposera sur l'utilisation d'une table de hachage où vous conserverez chaque mot associé à son score respectif. Vous pourrez vous servir de ce que vous avez réalisé lors des *labs* du module SD (et donc de la fonction naïve de hachage qui vous avait été fournie).

Les exécutions précédentes correspondent à l'attendu minimal de votre projet. Ce qui devrait vous permettre d'obtenir au plus la moyenne vis-à-vis de la notation sur cette partie (réalisation).

Il est attendu que votre implémentation soit modulaire, commentée et testée.

TRAVAUX COMPLÉMENTAIRES

En complément, nous souhaitons que vous réalisiez les travaux suivants. Nous vous invitons à écrire des programmes différents qui font usage des mêmes composants/modules logiciels.

Études statistiques des données

Nous vous invitons à étudier le corpus des données qui vous ont été fournies en calculant un certain nombre de statistiques descriptives et en réalisant des visualisations : quels sont les mots récurrents ? les moins présents ? autres ?

De manière similaire, nous vous invitons à étudier les données que vous avez dérivées -- mots avec leur score associé -- : quels sont les mots les plus positifs, les plus négatifs ? où se situe(nt) la/les frontières ? êtes-vous capables de qualifier les mots en 5 catégories (« très négatif », « plutôt négatif », « neutre », « plutôt positif », « très positif ») ? quelle est la répartition des mots selon leur score ?

Vous pouvez réfléchir à la manière de détecter et corriger certaines incohérences obtenues. Faites preuves d'esprit critique et d'une bonne analyse de vos résultats.

Mesure de la fiabilité de votre modèle de prédiction

Vous pouvez essayer de réaliser une validation de votre algorithme de prédiction. Pour cela, diviser votre jeu de données en deux sous-échantillons : (i) le premier sous-échantillon dit « sous-échantillon d'apprentissage » contiendra 60% à 75% des élements du jeu de données original ; (ii) le second « de test » contiendra le reste des éléments. Vous « entrainerez » votre modèle en utilisant le sous-échantillon d'apprentissage. Vous estimerez ensuite l'erreur en comparant le score prédit pour chaque élément de l'échantillon de test à sa note réelle.

Une autre approche consiste à utiliser une validation croisée du type « *k-fold cross-validation* ». Nous vous laissons vous renseigner par vous-même sur cette approche.

Prétraitements additionnels des données

Nous vous invitons à proposer et implémenter des traitements qui pourraient améliorer la qualité de vos résultats d'analyse de sentiments (par exemple non prise en compte des mots vides (*stop words*), insensibilité à la casse).

Étude relative à l'utilisation et la performance de votre table de hachage

Nous vous invitons à analyser l'usage de votre table de hachage en mêlant études statistiques des données et de leurs répartitions relativement à la fonction de hachage utilisée, et, mesures empiriques des temps d'accès (en insertion, consultation, mise à jour) à votre table au cours de l'exécution de votre programme.

Vous pouvez ainsi effectuer une analyse du « facteur de compression » (*load factor*) de votre table, proposer l'utilisation d'une autre fonction de hachage plus adaptée (réalisant une distribution uniforme des clefs).

Applications à d'autres jeux de données

Il pourrait être intéressant d'étudier d'autres jeux de données (et pourquoi pas en langue française cette fois-ci).

BONNES PRATIQUES EN GÉNIE LOGICIEL

Veillez à organiser votre dépôt en séparant les codes informatiques (sources, tests, binaires), des données, des documents, etc.

Vous fournirez un fichier `README.md` à la racine de votre dépôt `git` précisant l'objectif du projet, les contributeurs, décrivant l'organisation du dépôt et les principaux cas d'utilisation.

Votre code sera découpé de façon modulaire en séparant les responsabilités (*separation of concerns*). Il devra être commenté et indenté. Vous veillerez à choisir un maintenir une cohérence certaine sur les conventions de codage et de nommage.

Votre code devra être testé. Vous réaliserez et fournirez des tests unitaires les plus exhaustifs possibles pour l'ensemble de composants logiciels. Pour ce faire, vous utiliserez une librairie (laissée au choix) de tests unitaires. Vous chasserez les fuites mémoire en utilisant par exemple l'outil `valgrind`. Nous vous invitons fortement à compiler vos unités de compilation `C` en utilisant les paramètres classiques `-Wall`, `-pedantic`, et aussi `-fsanitize=address` qui permet d'inclure un outil de détection des bugs de corruption de mémoire tels que les dépassements de tampon ou les accès à une zone mémoire suspendue.

La construction (*build*) de vos logiciels sera automatisée (en fournissant a minima un `Makefile`, mais vous pouvez également viser la mise en place d'un pipeline d'intégration continue sur GitLab).

Faites bon usage de votre outil de gestion de versions `git`. Les noms des contributeurs devront être correctement configurés (`git config user.name / user.email`). Vous ferez bon usage du fichier `.gitignore` pour ne commiter que les fichiers sources et non les binaires ou les fichiers temporaires. Les messages de commit seront clairs et explicites (description succincte de ce qu'apporte le commit). Un commit ce doit d'être auto-suffisant et de n'avoir qu'un seul but (correction d'une erreur, ajout d'une fonctionnalité, etc.). Vous devez donc commiter le plus souvent possible et faire de « petits » commits.

QUELQUES INDICATEURS RAPIDES DE LA CORRECTION DE VOS CALCULS INITIAUX

Exemple de scores moyens associés à un mot

MOT	NOMBRE D'OCCURRENCES	SCORE
fantastic	13	2.8461
horrible	11	0.6366
ok	462	1.9502

Exemple de score prédit pour un commentaire

En considérant qu'un commentaire avec un score moyen supérieur à 2.01 est positif et qu'un commentaire avec un score inférieur à 1.99 est négatif.

COMMENTAIRE	SCORE	QUALIFICATIF
It made me want to poke out my eyeballs	1.9064	négatif

Exemple de mots qui contribuent négativement et positivement

Parmi les mots « terrible », « horrible », « ok », « refreshing », « formulaic », le mot le plus positif est « refreshing » avec un score de 3.3478 et le mot le plus négatif est « horrible » avec un score de 0.6363.