Antonin Schnyder, Sofia Boteva, Nicolas Bovet

EMODYSSEY - EMOTION ODYSSEY

INTRODUCTION

Le but de ce projet est d'analyser l'œuvre de l'Odyssée et d'effectuer des analyses émotionnelles sur le texte afin de dégager les émotions des différents passages, puis de comparer les résultats entre différents systèmes. Le but est aussi de comparer plusieurs systèmes d'annotations d'émotions. Le même texte sera donc passé sous trois systèmes analysés différents afin d'en dégager les émotions associées. Le premier système est une extraction manuelle des émotions faite par des humains, le deuxième système utilise un modèle nommé Senticnet pour extraire les émotions et le dernier système repose sur un module d'extraction des émotions nommé PyFeel qui repose sur un traitement en langage naturel. Les résultats permettront de déterminer s'il y a une bonne ou mauvaise concordance entre les différents systèmes et si la détection est fiable ou non.

Les données brutes sont des fichiers au format json dans lesquels sont stockées les répliques de l'Odyssée et qui ont été taguées manuellement par des humains. Les fichiers sont répartis dans 13 dossiers différents (qui correspondent à 13 personnes différentes). Chaque fichier est un extrait qui a été tagué par un participant. Chaque participant a tagué 4 passages (parmi 24 extraits au total), dont certains se retrouvent parmi d'autres participants. Chaque passage a été tagué par une émotion parmi 7 (calmness, angry, surprise, faer, disgust, sadness, joy) sur le site de Tagtog (http://tagtog.com). Les émotions associées à chaque dialogue ont été stockées dans un dataframe avec le passage associé. Le passage sert aussi de base pour les analyses émotionnelles avec Senticnet et PyFeel telles que décrites ci-dessous.

IMPORTATION DES DONNÉES ET STATISTIQUES

Pour extraire les données des JSON, traiter celles-ci en évaluant les textes tagués avec les méthodes Senticnet et PyFeel, et enfin en tirer des statistiques, la classe EvalFeel a été créée. Celle-ci s'initialise à partir d'un chemin indiquant le dossier dans lequel sont stockés les fichiers JSON des textes tagués avec TagTog. Elle charge ces fichiers et les rassemble dans une seule dataframe qu'elle enregistre dans un fichier csv. EvalFeel a ensuite pour but d'effectuer ces différentes opérations au moyen de ses fonctions que nous allons détailler ci-dessous.

La fonction make_df a pour but de former une dataframe à partir des fichiers JSON bruts qui vient de la base de données TagTog. Ces fichiers contiennent tous les extraits tagués avec une émotion par une série d'utilisateurs différents. Ce qui nous intéresse dans ces JSON est essentiellement l'extrait et l'émotion correspondante. Ainsi la fonction *make_df* garde ces deux informations pour former une nouvelle dataframe tout en nettoyant les lignes jugées non pertinentes, soit les lignes avec un extrait ne comportant qu'un seul mot. En effet, ces lignes sont considérées indésirables, car elles semblent être toujours des erreurs humaines lors du tag, car le texte est par exemple composé d'une seule lettre ou

d'un mot propre donc d'un orateur et non d'une émotion exprimée dans un dialogue. Ainsi la dataframe nouvellement obtenue est stockée dans un fichier csv.

La fonction *add_emotions_senticnet* est chargée d'ajouter une émotion grâce à l'algorithme de la librairie senticnet correspondant à chaque extrait de texte tagué par les utilisateurs sauvegardé dans la dataframe fabriqué par la fonction *make_df*. Pour cela il faut donner une liste de mots à la fonction *averageEmotionsOf* de la librairie senticnet qui se charge d'évaluer l'émotion moyenne correspondante à la liste donnée. Cette liste reprend successivement chaque mot des extraits excepté ceux qui n'ont pas d'intérêt pour une évaluation émotive, ces mots sont généralement appelés les « stopwords ». Une fois les stopwords retirés, les extraits peuvent donc être découpés en listes de mots et donnés à la librairie senticnet. Les émotions retenues pour chaque extrait sont ajoutées dans une colonne de la dataframe de base.

La fonction *add_emotions_pyfeel* est chargée d'ajouter une émotion grâce à l'algorithme de la librairie pyfeel correspondant à chaque extrait de texte tagué par les utilisateurs sauvegardé dans la dataframe fabriqué par la fonction *make_df*. Pour cela, il faut donner une chaine de caractères à la fonction *Feel* de la librairie pyfeel qui se charge d'évaluer l'émotion moyenne correspondante à la chaîne. Cette fonction ne retourne pas une seule émotion moyenne comme la librairie senticnet mais un score pour chaque émotion évaluée par pyfeel. Il s'agit donc dans notre cas pour faire au plus simple de retenir l'émotion avec le score le plus haut. Le cas particulier de ce processus est l'émotion « positivité » que pyfeel considère comme une émotion et reçoit un score au même titre que les autres. Dans les faits, « positivité » indique une tendance d'une chaine de caractère à comprendre un sens plus ou moins positif, cela n'est donc pas une émotion pertinente utile au but de comparatif de ce travail, nous avons donc opté pour l'écarter des résultats quand il fallait choisir le score de l'émotion le plus haut. Cependant, nous avons décidé que lorsque toutes les émotions ont un score nul et que « positivité » est positif, l'émotion « calmeness » est attribuée, car celle-ci n'est pas dans le panel de base, mais peut correspondre à un extrait pas spécialement émotif et dont la température serait plutôt positive. Les émotions retenues pour chaque extrait sont ajoutées dans une colonne de la dataframe de base.

La fonction *make_diagram* est chargée de générer un diagramme de Venn à partir d'une dataframe et d'une émotion donnée en paramètre. Ainsi, elle génère trois différents ensembles basés sur les occurrences de l'émotion donnée en paramètre dans les colonnes TagTog', 'PyFeel' et 'Senticnet'. Cela donne le nombre d'extraits tagués avec l'émotion donnée en paramètre. Ensuite quatre sous-ensembles sont formés à partir des extraits dont la même émotion a été taguée par deux ou trois méthodes différentes. Ainsi nous obtenons en plus les extraits avec des émotions en commun dans au minimum deux méthodes et au maximum les trois méthodes. Ensuite un diagramme de Venn est créé à partir des trois ensembles globaux de chaque méthode et les intersections entre chacune de ces méthodes. Une dataframe est également créée à partir de ces données est retournée,

cela permet de garder ces résultats obtenus pour une émotion en paramètre, et donc pouvoir agréger plusieurs itérations de cette fonction utilisée pour plusieurs émotions.

La fonction *make_all_stats* est la fonction qui a pour objectif de rassembler toutes les statistiques de chaque émotion présente dans la dataframe de base et en sortir leur diagramme de Venn correspondant. Pour ce faire nous pouvons décider de normaliser la dataframe avec les émotions et les textes afin qu'il y ait le plus d'émotions avec la même nomenclature à comparer. Une fois cette dataframe normalisée ou non, on parcours chaque émotion présente et on appelle la fonction *make_diagram* qui permet de rassembler toutes les statistiques et de créer à chaque émotion un diagram stocké au format png.

RÉSULTATS

Les résultats obtenus grâce à la fonction *make_all_stats* de la classe **EvalFeel** ont pour nature d'indiquer le nombre d'extraits tagués par une méthode pour chaque émotion d'une part, et d'autre part d'indiquer les similarités entre chaque méthode pour chaque émotion. Ces statistiques sont produites sous forme de dataframe qui rassemble les résultats de chaque émotion et un diagramme de Venn par émotion. Le diagramme de Venn à l'avantage d'indiquer de manière intuitive la quantité d'extraits pour chaque méthode grâce à la taille des cercles et aussi les extraits taggés avec la même émotion par plusieurs méthodes grâce à la taille des intersections.

Les résultats obtenus indiquent les différentes tendances de chaque méthode. Aucune émotion ne fait l'unanimité au niveau de sa similarité entre méthodes, l'intersection des trois étant toujours de taille modeste. Pour un travail ultérieur qui voudrait se pencher sur l'analyse en détails de ces résultats, il convient de rappeler quelques limites à notre processus.

D'abord le choix de normaliser le nom des émotions taguées par senticnet peut se discuter. Bien que nous l'ayons fait en se basant sur le schéma de correspondance fait par l'équipe de senticnet (https://sentic.net/about/), cela reste une réduction d'informations et donc une perte de précisions. Il en va de même avec le choix de sélectionner uniquement l'émotion avec le score le plus élevé donné par la méthode PyFeel : il est au vue de notre processus puisque nous pouvons retenir qu'une émotion par méthode et par extrait, mais nous perdons de l'informations qui peut être utilisée, surtout dans le cas de scores proches. Enfin, l'utilisation de la libraire senticnet n'exploite pas ses pleines capacités. En effet, cette librairie évalue des émotions en prenant en entrée des groupes de mots aussi bien que des mots uniques. Or, dans la classe EvalFell, nous n'utilisons que des mots uniques, la raison de ce choix est portée par le temps qu'il faudrait pour découper chaque extrait en potentiel groupe de mots reconnus par l'immense dictionnaire de senticnet. Nous perdons encore là un peu de précision dans l'évaluation des textes par la librairie.

ONTOLOGIE

Pour construire la base de notre ontologie, nous avons utilisé l'ontologie sur les émotions et les profils des personnages littéraires *"psy_model.owl"*. Nous avons ensuite apporté des modifications en fonction de nos objectifs et nos besoins.

L'objectif de cette ontologie est de représenter la catégorisation des émotions selon trois méthodes d'étiquetage : Tagtog (étiquetage manuel), Senticnet et Pyfeel, ainsi que d'ajouter des statistiques relatives à la détection des émotions dans les speechs et garder en mémoire les résultats de cette détection (construction de la A-box).

T-BOX / CLASSES

Pour la construction de T-box nous avons ajouté une sous-classe à la classe Emotion, nommée EmotionProgramme. Cette classe est composée de trois sous-classes correspondant aux trois méthodes d'étiquetage. Chaque classe d'émotion par programme possède ses propres sous-classes représentant les émotions distinguées par chaque outil. Afin d'établir une corrélation entre les émotions distinguées par les différentes méthodes, chaque émotion des trois programmes a comme classe-mère une des émotions de base (BasicEmotion) de l'ontologie initiale.

Pour faire la comparaison entre les systèmes émotionnels nous avons pris comme référence les systèmes de Pyfeel et Tagtog qui sont déjà suffisamment similaires entre eux (sauf l'émotion "calmness", présente dans tagtog mais absente dans Pyfeel). Nous avons dû ensuite réajuster le système de Senticnet qui propose une analyse des émotions plus vaste et nuancée par rapport aux deux autres méthodes, pour que les émotions des trois méthodes s'allignes entre elles.

Ainsi, nous avons décidé de représenter les émotions de Senticnet d'une manière hiérarchique. Nous avons pris pour base le sablier émotionnel de Senticnet et nous l'avons réajusté suivant nous besoins. Dans l'ontologie finale chaque émotion de Senticnet est représentée par une classe-mère et deux classes-filles (par exemple, JoySenticnet a pour classes-filles Contentment et Ecstasy). Comme classe mère de chaque émotion nous avons pris l'émotion centrale du schéma, qui est la plus "neutre" et qui correspond aux émotions de Tagtog et Pyfeel, tandis que les émotions filles sont représentées par les émotions "extrêmes".

La plupart des émotions principales des trois méthodes coïncident, mais il existe des exceptions :
- L'émotion "calmness" n'existe pas dans la classification de Pyfeel.
- L'émotion "eagerness" (y inclus ses sous-classes) n'existe que dans la classification de Senticnet.
- L'émotion "surprise" n'est pas distinguée dans Senticnet, nous avons donc utilisé "Pleasentness" comme équivalent.

Nous avons également réorganisé les émotions de base de l'ontologie initiale pour assurer une meilleure cohérence entre les émotions de base et les émotions des trois systèmes émotionnels:
- Dans les émotions de base nous avons ajouté les émotions "Calmness" et "Eagerness" qui n'étaient pas présentes initialement.
- Certains noms d'émotions de base ont également été modifiés pour assurer une meilleure cohérence entre les émotions de base et les émotions des trois systèmes émotionnels (l'émotion-mère "Terror" a été renommée en "Fear" et l'émotion-fille "Fear" a été renommée en "Terror").

A-BOX / PROPRIÉTÉS
Pour la création de la A-Box, un certain nombre de propriétés ont dû être ajoutées à l'ontologie pour satisfaire les besoins.

Trois "datatype property" ont été créées pour satisfaire les besoins de la modélisation : (1) ***hasTotalEmotions***, qui permet d'exprimer le nombre total d'émotions détectées sur d'un certain type sur l'ensemble des dialogues, (2) ***hasAllCommonEmotions***, qui permet d'exprimer le nombre de fois où les trois systèmes d'émotions ont trouvé la même émotion et coïncident, (3) ***hasCommonEmotion***, qui permet d'exprimer le nombre d'occurences communes lorsqu'on compare 2 systèmes d'émotions. Ces trois propriétés seront utiles lors de la création de la A-Box (ci-dessous). Toutes les propriétés ajoutées sont fonctionnelles.

Trois "object property" ont été créées pour satisfaire les besoins de la modélisation : (1) ***hasTagtogEmotion***, (2) ***hasSenticnetEmotion*** et (3) ***hasPyfeelEmotion***. Elles permettent de lier un speech ou une statistique à une émotion selon l'un des trois systèmes étudiés. Un speech peut ainsi être lié à une émotion de Joy dans TagTog, Sadness dans SenticNet et Sadness dans Pyfeel. Ces propriétés sont aussi toutes fonctionnelles et seront utiles lors de la création de la A-Box (ci-dessous).

Pour l'ajout des individus à l'ontologie (création de la A-Box), on a complété les étapes suivantes. L'ontologie utilisée comme base est *"psy_model.owl"* qui a été modifiée. Le module Python utilisé pour créer la A-Box est nommé *abox.py* et repose sur la bibliothèque rdflib. L'ontologie modifiée et contenant la A-Box est exportée sous le nom de *"psy_model_filled.rdf"*.

La A-Box est constituée de plusieurs éléments. Le premier élément est les statistiques relatives aux différents programmes de détection des émotions. Les données sont générées par le programme tel que détaillé ci-dessus et réimportées dans le programme grâce à la méthode loadStats(). L'ajout des individus à l'ontologie se fait grâce à la méthode fill(). La première étape est d'associer le nombre d'occurence trouvée pour chaque émotion. La distinction se fait selon chaque système. Cela signifie qu'on entre dans la A-Box le nombre d'émotions Joy détectées par TagTog, SenticNet et PyFeel, par

exemple. On fait de même avec toutes les autres émotions. Ces ajouts se font grâce aux propriétés ***hasTotalEmotions***, décrite ci-dessus.

On ajoute aussi à la A-Box le nombre d'occurences où les trois système ont détecté la même émotion pour un même speech. Cet ajout se fait au travers de la propriété ***hasAllCommonEmotions*** en tant qu'entier positif ou nul. Finalement, le dernier type de statistiques se fait en comparant les différents systèmes sur toutes les émotions. À chaque fois, une émotion de base (par exemple Joy) est comparée une première fois entre Tagtog et PyFeel, ensuite entre TagTog et Senticnet et finalement entre Senticnet et PyFeel. La A-Box est constituée en créant un identifiant de statistique unique à chaque comparaison, auquel on ajoute les émotions qui sont comparées en utilisant les propriétés hasTagtogEmotion, hasSenticnetEmotion et hasPyfeelEmotion selon la comparaison effectuée. Le nombre d'occurence commune est donné au travers de la propriété ***hasCommonEmotion*** et prend la forme d'un entier positif ou nul.

Le second élément de la A-Box est les données des speech et les émotions associées. L'ajout des individus à l'ontologie se fait grâce à la méthode fill(). Les données sont générées par le programme tel que détaillé ci-dessus et réimportées dans le programme grâce à la méthode loadData(). Chaque speech est ajouté à l'ontologie en tant qu'individu de type speech/string. L'id de chaque speech correspond à son numéro dans le dataframe importé, de telle sorte qu'il n'y ait aucun duplicat. Les propriétés hasTagtogEmotion, hasSenticnetEmotion et hasPyfeelEmotion permettent de lier les émotions détectées par chacun des différents programmes au speech. Ceci permet de garder une trace des différentes émotions détecter par chaque programme. Finalement, on ajoute aussi le contenu du speech au travers de la propriété dc_subject, déjà présente dans l'ontologie, dans le but de compléter la A-Box de manière exhaustive.

La A-Box ainsi constituée permet de garder une trace des résultats de chaque système sur la détection des émotions ainsi que quelques statistiques descriptives de base.