TRAITEMENT AUTOMATIQUE DES LANGUES

PROJET : Analyse d'opinions dans Twitter

Rafael Pallares
Sall Papa Amadou
Odon Rochefort

Responsable

Delphine Bernhard

http://sites.google.com/site/delphinebernhard

INTRODUCTION

Les utilisateurs de Twitter expriment régulièrement leur opinion sur une diversité de thématiques. Ces informations subjectives permettent de savoir, de manière quasi-instantanée, ce que les personnes pensent à propos d'un sujet donné.
L'analyse automatique des opinions est trés utile pour les entreprises, qui veulent savoir ce que leurs clients pensent d'un produit et pour les personnes individuelles, qui veulent savoir ce que d'autres pensent d'un sujet (politique, etc.) ou d'un produit (livre, film, voiture).
L'objectif de ce proget est développer une application qui, pour un sujet donné, est capable d'extraire les opinions exprimés à partir de l'application Twitter.
Nous allons dans un premier temps constituer un corpus de tweets en français sur un sujet donné à l'aide de l' API Java. Twitter ,comme presque tous les réseaux sociaux, est un forum de rencontre où les personnes s'expriment librement sans ce soucier de l'orthographe des mots ou de la syntaxe grammaticale des phrases. La deuxième étape sera donc de faire une correction orthographique sur notre corpus afin d'obtenir un texte significatif sur lequel il sera plus facile de faire nos traitements.
Avant d'entamer notre phase de traitement d'opinion, nous allons constituer un lexique composé d'adjectifs, à polarité négative ou positive, caractéristique du langage subjectif permettant d'exprimer des opinions, puis l'enrichir.
Nous terminerons, par la phase d'extraction et d'analyse d'opinions , puis la présentation des statistiques sur le sujet donné.

CONSTITUTION DU CORPUS

La classe « MakeCorpus »

Cette classe nous permet, à partir de l'API Twitter ,de collecter des tweets après une requête.

La requete est le sujet sur lequel on veut récupérer ces tweets.

(Ex: “Équipe de France”, “révolution”, “Sarkozy”)

Cette classe définie une classe interne “`makeRequete`”, qui à partir une chaine entrée, construit une requête qu'elle passe ensuite à l'API Twitter pour extraire les tweets.

```
private static String makeRequete () {
                  List<String> verbs = Opinion.loadVerbs();
                  verbs.addAll(Lexique.initNegatif());
                  verbs.addAll(Lexique.initPositif());
                  Collections.shuffle(verbs);
                  int max = 30;
                  String req = "";
                  for(int i = 0; i<max && i<verbs.size();i++) {
                        String s = verbs.get(i);
                        req += s + " OR ";
                  }

                  req = req.substring(0, req.length()-3);

                        req += "et OR mais ";
                  return req;
}
```

Nous avons créé cette classe afin de maximiser les chances d'obtenir des messages pertinents. En effet la plupart des messages twitter ne contiennent aucun contenu analysable, le fait de forcer la requête à contenir au moins un des mots du lexiques, et de contenir une des conjonctions de coordinations que nous sommes en mesure d'analyser, nous permet d'exploiter une plus grande partie des messages reçus.

La classe « FormatCorpus »

Cette classe permet de “normaliser” les tweets. En d'autres termes, elle permet de redéfinir le format des tweets en supprimant le maximum d' éléments qui peuvent être source d'ambiguïté durant notre traitement. Comme les accents, les majuscules, quelques signes de ponctuations n'appartenant pas a la langue courante.

La classe « TwitterSpellChecker »

Cette classe permet de faire la correction orthographique à l'aide de l'outil hunspell.

Les termes utilisés dans les tweets sont très semblable à ceux du langage sms, d'où la nécessité de les corriger. La classe charge l'ensemble des tweets et procède à la correction orthographique des mots. Lorsque plusieurs corrections possibles sont proposées, on utilise systématiquement la première de la liste.

Cependant, certaines propositions ne devraient pas être prises en compte, car ce ne sont pas des fautes.

C'est le cas des pseudo et des mots qui se trouvent pas dans le dictionnaire français mais couramment utilisés dans la langue française. Cette dernière catégorie de mot usuels se trouvant pas dans le dictionnaire sont appelés des “faux positifs”(« lol », « mdr » et autres). Pour éviter que ces mots soient considérés comme des fautes, nous les ajoutons à notre dictionnaire.

Analyse du corpus

La classe « Lexique »

Comme annoncé dans l'introduction, nous allons construire un lexique affectif comprenant des mots et des expression caractéristique du langage subjectif ainsi que leur orientation (positif ou négatif).

Nous allons partir d'un lexique initial, constituer d'adjectifs couramment utilisés et qui permettent de reconnaitre la polarité d'une expression, soit positif ou négatif.

A partir donc de deux listes d'adjectifs, de polarité positive d'une part et négative d'autre part et à l'aide de patrons, nous allons extraire automatiquement tous les adjectifs coordonnées à nos adjectifs initiaux.

Mais avant, nous il a fallu générer les arbres syntaxiques sur lesquels nous allons utiliser ces patrons.

Exemples de patrons :

```
ADJ=adj1 . (COORD < (CC[<< et] . (AP < ADJ=adj2)))
ADJ=adj1 . (COORD < (CC[<< et] . (AP < (ADV . ADJ=adj2))))
ADJ=adj1 . (COORD < (CC[<< et] . (VN . (AP < ADJ=adj2))) )
AP < (ADV . ADJ=adj1) . (COORD < (CC[<< mais] . (AP < ADJ=adj2)))
AP < (ADV . ADJ=adj1) . (COORD < (CC[<< mais] . (AP < ADJ=adj2)))
ADJ=adj1 . (COORD < (CC[<< mais] . (VN . (AP < ADJ=adj2))) )
```

Chaque patron est stocké dans une liste de Doublets qui contient le patron ainsi que la polarisation des deux adjectifs qui sont reconnus par chaque patron. Par exemple le patron « ADJ . Et . ADJ » a une polarisation vraie. C'est a dire que si un des deux adjectifs est rencontré, il appartiendra a la même polarité que l'autre. A contrario, les patrons formés a base de « mais » on une polarisation fausse, permettant de reconnaitre deux adjectifs de polarités inverses. Grâce a cette structure il est aisé de rajouter des patrons. Car plus il y aura de patrons pertinents, plus le lexique s'enrichira.

Finalement il peu arriver que des adjectifs se retrouvent dans les deux polarités à la fin de l'algorithme. On supprime donc les adjectifs qui sont présents dans les deux listes de polarités.

Afin de maximiser nos chances de reconnaitre des adjectifs, le parcours du corpus est effectué récursivement. En effet si un adjectif à été reconnu a la fin du fichier et ce dernier ajouté, il se peut qu'il ait été ignoré lors de la première passe. On ré-effectue donc le traitement tant que les listes d'adjectifs n'ont pas changés lors d'une passe.

La classe « Opinion »

La classe Opinion se charge d'analyser si un tweet est subjectif ou non. Et s'il l'est de déterminer si ce dernier exprime plutôt une opinion positive ou négative.

Cette analyse se fait en un seul parcours de tweets. On dispose premièrement d'une base de verbe exprimant une opinion. Pour chaque verbe expressif contenu dans le tweet, la variable d'expression d'opinion est incrémentée.

Parallèlement on calcule si le tweet exprime une opinion positive ou non. Pour ce faire, on va chercher dans la phrase les mots de notre lexique affectif. Suivant la polarité de l'adjectif trouvé, on incrémente les variables d'expression négative ou positives.

Enfin l'analyse la plus expressive est le parcours des patrons que nous avons dans notre base. Chaque patron est associé à un entier donnant des informations sur le patron en question.

```
/**
        * retourne la liste des doublets des patrons d'opinions
        * Si n>0 <==> nb adjectifs a analyser
        * si n==0 patron positif
        * si n==-1 patron negatif
        * si n==-2 mixte
        * @return
        */
       public static List<Doublet<String, Integer>> loadSubjPatron () {
```

Suivant l'entier associé au patron, on est capable de savoir le nombre d'adjectifs que reconnais le patrons (afin de polariser d'autant plus ce dernier), si le patron exprime une opinion négative, positive, ou mixte. Nous avons rajouté l'expression mixte dans un patron car nous considérons que ces derniers sont toujours porteurs d'opinions mais qu'il n'est pas toujours possible de définir si elle est positive ou négative. Nous espérons qu'un autre patron ainsi que les adjectifs affectifs nous renseigneront a ce sujet. Au final lorsqu'un tweet est reconnu par un patron et qui contient un ou plusieurs adjectifs de notre lexique, le tweet est polarisé de manière plus importante et est considéré comme porteur de plus d'opinion qu'un simple verbe d'opinion et/ou d'adjectif pondérant.

Cette structure de donnée permet d'enrichir facilement notre base de patrons. Cette méthode nous a permis d'obtenir des résultats de plus en plus satisfaisant a mesure que l'ont rajoute des patrons.

Au final le tweet est classé selon deux critères, sa subjectivité puis sa pondération. Ainsi un tweet fortement pondéré mais non porteur d'opinion ne sera pas expressif. Pour dire si un tweet est porteur d'opinion, on comparera si son score de subjectivité est supérieur à un certain seuil.

Enfin afin de définir si un tweet est positif ou négatif, on prend le maximum des score positifs et négatifs. Il est probable que ce choix ne soit pas optimal, il faudrait surement avoir une méthode plus fine en prenant en compte le nombre d'expression d'affection total ou la distance entre le nombre de positifs ou négatifs. Mais les tweets étant des messages très court, notre solution reste satisfaisante.

Affichage des statistiques

Lors de l'analyse des tweets on compte le nombre total de tweets, le nombre de tweets subjectifs et le nombre de tweets positifs/négatifs/mixtes.

Ces données permettent de comprendre assez bien la tendance de la requête de départ. Bien que beaucoup de tweets ne soient pas ou très peu expressifs, et malgré les erreurs commises par la classification, la tendance générale pour des sujets connus correspond assez bien à la réalité. On peu considérer que plus le nombre de tweets est important, plus la réponse sera juste, c'est pourquoi d'ailleurs lors de la récupération des tweets, plusieurs appels a Twitter sont effectués.

De plus, avant l'affichage final des statistiques, nous affichons les adjectifs trouvés par les patrons d'opinions. Ainsi lorsqu'un adjectif inconnu est trouvé, il nous est possible d'enrichir manuellement notre lexique en définissant si ce dernier exprime une opinion positive ou négative.

Exemples de sorties du programme :

Séquence d'exécution:

Il est à noter qu'afin de rendre le programme plus interactif, il serait judicieux d'effectuer cette séquence dans un script bash.

```
MakeCorpus.java

TwitterSpellChecker.java

Format_corpus.java

$BONSAI/bin/bonsai_bky_parse_via_clust.sh corpus_correct.txt > arbres.txt

Lexique.java

Opinion.java
```

```
Requête = bieber
Total tweets : 403, dont 114 sont subjectifs.
     Tweets positifs : 62
     Tweets négatifs : 27
     Tweets mixtes : 25
```

```
Requête = sarkozy
Total tweets : 146, dont 64 sont subjectifs.
     Tweets positifs : 10
     Tweets négatifs : 38
     Tweets mixtes : 16
```

```
Requête = ipad
Total tweets : 152, dont 35 sont subjectifs.
     Tweets positifs : 6
     Tweets négatifs : 4
     Tweets mixtes : 25
```

```
Requête = revolution
Total tweets : 137, dont 58 sont subjectifs.
     Tweets positifs : 24
     Tweets négatifs : 3
     Tweets mixtes : 31
```