ARNAUD SOULIER

MATHIEU REMY

POUR LE 04/05/2018

PROJET TALN : COMPTE-RENDU FINAL

Analyse de sentiment et génération de tweets

PROBLEMATIQUES

L'analyse de sentiment est un champ du traitement du langage naturel, qui consiste à déterminer la tonalité émotionnelle d'un mot, une phrase ou un texte. Cette tonalité peut être plus ou moins négative ou positive, mais également plus ou moins objective ou subjective.

D'autre part, l'enjeu de la génération de tweets est de créer des phrases courtes bien formées et porteuses de sens, dans le but d'être par la suite publiées sur Twitter.

APPROCHE CHOISIE

La première partie de notre projet consistera à créer une application qui, à partir d'un sujet (mot-clé), ira récupérer des tweets, pour réaliser une analyse d'opinions : c'est-à-dire déterminer si un sujet est perçu positivement ou négativement en ce moment par les internautes, par exemple en lui assignant une note.

Dans la seconde partie, nous créerons un module qui ira chercher la tendance (hashtag) en top-tweet la plus positive du moment, pour générer un tweet positif à son sujet et le mettre en ligne.

L'objectif final est donc de concevoir un agent intelligent qui cherchera à suivre les tendances et à se conformer relativement aux autres sur Twitter.

TECHNIQUES EXISTANTES

Cette bibliographie rassemble diverses informations, modules et outils qui pourront s'avérer utiles dans la conception de notre projet. Nous avons principalement accès notre recherche sur le traitement de l'anglais (mais pas que) et le langage Python, qui dispose d'un large panel de modules pour les domaines qui nous intéressent.

RECUPERATION ET PUBLICATION DE TWEETS

- **Twitter developer**: https://developer.twitter.com/en.html
  Site web officiel où il faudra de créer un compte développeur, qui conditionne l'accès aux API (car nécessité de s'authentifier sur le réseau pour accéder aux données). Nous utiliserons le service gratuit, qui fournit un accès limité aux données, mais qui s'avère suffisant ici.
- **Tweepy**: http://www.tweepy.org/
  C'est une API python de Twitter populaire et user-friendly. Elle sera utile pour interagir avec Twitter : récupérer des stream de tweets, la trend list du moment, mettre en ligne un tweet, etc.
- **Twitter-python**: https://github.com/bear/python-twitter
  Une alternative à l'API précédente.

ANALYSE DE SENTIMENT

- **NLTK**: https://www.nltk.org/
  Une boîte à outils complète pour Python (voir **TextBlob**).
- **TextBlob**: http://textblob.readthedocs.io/en/dev/index.html
  Une autre boîte à outils, basée sur NLTK, davantage user-friendly. Elle contient comme fonctionnalités : **des dictionnaires de toute sorte (dont WordNet), le parsing, les n-grammes, l'extraction de groupes nominaux, les principaux classifieurs (naïve bayésienne, arbre de décision, etc.), la traduction, la tokenisation**, et bien d'autres choses.
- **Vader**: https://github.com/cjhutto/vaderSentiment
  Un module et lexique adapté à l'analyse de sentiment sur les réseaux sociaux.
- **LVF**: http://rali.iro.umontreal.ca/rali/?q=fr/lvf
  Un dictionnaire des verbes français.
- **FondamenTAL** : http://talep.lif.univ-mrs.fr/FondamenTAL/
  Une page listant de nombreuses ressources en français.
- **Dicovalence**: https://www.ortolang.fr/market/lexicons/dicovalence
  Un autre dictionnaire des verbes français.
- **FEEL**: http://www.lirmm.fr/~abdaoui/FEEL.html
  Un dictionnaire recensant les polarités de 14000 mots distincts en français.

- **WordNet**: http://wordnetweb.princeton.edu/perl/webwn
  Une base de données lexicale où les mots sont triés en « synsets ».
- **SentiWordNet**: http://sentiwordnet.isti.cnr.it
  Extension de WordNet ajoutant leur polarité aux synsets.
- **JeuxDeMots** : http://www.jeuxdemots.org/jdm-accueil.php
  Un dictionnaire d'association lexicale en français.

GENERATION DE TWEETS

- **NLTK** : voir plus haut.
- **Chaînes de Markov** : https://fr.wikipedia.org/wiki/Cha%C3%AEne_de_Markov
  Un article Wikipédia sur cet outil mathématique qui pourrait être utile.
- **NumPy**: http://www.numpy.org/
  Un module efficace pour la gestion des matrices en Python.
- **TFLearn**: http://tflearn.org/
  Un module de Machine Learning basé sur **TensorFlow** et plus user-friendly que ce dernier. L'ayant déjà utilisé par le passé, nous l'envisageons comme une piste intéressante pour la génération de textes.

COMPARAISON DES OUTILS EXISTANTS

Cette section rend à la fois compte des articles que nous avons consultés lors de la recherche d'informations, ainsi que des projets similaires existants dans le domaine. Certains pourraient sembler redondants à première vue, mais nous n'avons listés que ceux qui nous ont paru significatifs (présence d'une ou plusieurs informations inspirantes ou utiles).

ANALYSE DE SENTIMENT

- **Analyse de sentiment Twitter** : http://www.agroparistech.fr/ufr-info/membres/cornuejols/Teaching/Master-AIC/PROJETS-M2-AIC/PROJETS-2016-2017/analyse-de-sentiments(lambert-bellard-lorre-kouki).pdf
  Un article résumant assez bien l'état de la recherche.
- **Sentiment140** : http://www.sentiment140.com/
  Un logiciel d'analyse de sentiment sur Twitter relatif à des sujets, des produits ou des marques. Il utilise des algorithmes de Machine Learning (plus précisément : la Distant Supervision).

- **ResTS**: http://www.aclweb.org/anthology/F12-3018
  Un article à propos du logiciel ResTS, qui réalise des résumés d'opinions sur des produits commerciaux à partir de Twitter et **SentiWordnet**. On y trouve un descriptif détaillé des techniques employées (équations mathématiques et pseudo-codes).
- **Best free tools**: https://www.softwareadvice.com/resources/free-twitter-sentiment-analysis-tools/
  Un article présentant une pléiade de logiciels d'analyse de sentiment sur Twitter.
- **Sentiment analysis in Python** : http://blog.aylien.com/build-a-sentiment-analysis-tool-for-twitter-with-this-simple-python-script/
  Un article décrivant une implémentation en Python.
- **Step-by-Step Twitter Sentiment Analysis**: http://ipullrank.com/step-step-twitter-sentiment-analysis-visualizing-united-airlines-pr-crisis/
  Un autre article d'analyse de sentiment sur Twitter, présentant une implémentation utilisant la classification naïve bayésienne (du module NLTK).
- **Twitter Sentiment Analysis using Python**: https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/
  Une autre implémentation, utilisant **TextBlob**.
- **Azure Stream Analysis**: https://docs.microsoft.com/fr-fr/azure/stream-analytics/stream-analytics-twitter-sentiment-analysis-trends
  Un article sur l'analyse de sentiment avec Microsoft Azure, la plateforme de cloud-computing au succès grandissant (dispensable mais intéressant).
- **Sentiment Viz** : https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/tweet_app/
  Un autre exemple logiciel (avec une interface).
- **Another Twitter sentiment analysis with Python** :
  https://towardsdatascience.com/another-twitter-sentiment-analysis-bb5b01ebad90
  Un "article-fleuve" en 11 parties présentant de très nombreuses techniques.
- **How to Build the Trump Twitter Sentiment Analysis Dashboard**:
  https://hackernoon.com/visualizing-trump-7e1cb5e55a78
  Un analyseur de sentiment utilisant **PubNub** et **IBM Watson**.
- **Sentiment Analysis of Today's Tweets**:
  https://medium.com/@olafusimichael/sentiment-analysis-of-todays-tweets-about-president-buhari-using-python-s-vader-and-sentiwordnet-c4a42e8c748f
  Utilisation intéressante de **Vader** et **SentiWordNet**.
- **ADVANSE**: http://talnarchives.atala.org/ateliers/2015/DEFT/deft-2015-long-009.pdf
  Un article complet sur un analyseur de tweets français.

GENERATION DE TWEETS

- **Génération automatique de textes** : https://interstices.info/jcms/int_63524/generation-automatique-de-textes
  Un article général sur le sujet.
- **Conversational UI Principles** : https://medium.com/swlh/conversational-ui-principles-complete-process-of-designing-a-website-chatbot-d0c2a5fee376
  Un article sur le fonctionnement des chatbots.

- **Génération de textes**: https://en.wikipedia.org/wiki/Natural_language_generation et https://fr.wikipedia.org/wiki/G%C3%A9n%C3%A9ration_automatique_de_textes
  Les articles Wikipédia sur le sujet.
- **SimpleNLG** : https://github.com/simplenlg/simplenlg
  Un générateur de textes réalisé en Java, intéressant à décortiquer.
- **SimpleNLG-EnFr** : http://www-etud.iro.umontreal.ca/~vaudrypl/snlgbil/snlgEnFr_english.html
  Extension de SimpleNLG pour le français.
- **poetry-generator**: https://github.com/schollz/poetry-generator
  Un générateur de poèmes.
- **Générateur automatique de textes** : http://www.samszo.univ-paris8.fr/Generateur-automatique-de-textes
  Un projet universitaire achevé.
- **Natural Language Generation :** http://www.inf.ed.ac.uk/teaching/courses/nlg/
  Un cours universitaire.

CAHIER DES CHARGES DU PROTOTYPE

A priori, nous allons réaliser notre prototype en Python, étant donné la variété de modules disponibles dans ce langage. Le projet étant ambitieux et divisé en deux parties, Python nous permettra de passer plus de temps sur la partie algorithmique, plutôt que de s'enliser dans des problématiques de récupération ou de structuration de données. Par ailleurs, nous ne nous intéresserons qu'à la partie anglophone de Twitter.

L'utilisation du programme se fera directement dans la console, sauf si nous avons suffisamment de temps pour réaliser une petite interface graphique très minimaliste et modeste. Nous avons conscience de l'étendu du travail qu'il faudra réaliser, mais nous sommes déterminés à le mener à son terme.

DONNEES

- **Twitter** : La première tâche à réaliser sera d'implémenter la récupération de n tweets à partir d'un mot-clé au moyen d'une API. Ensuite nous allons devoir filtrer (**stop-words**) et structurer ces données (**TextBlob**) pour l'usage qui va en être fait. Il faudra également que notre programme soit capable de récupérer la trend-list du moment, et aussi de publier un simple tweet.

- **Les dictionnaires** : Le choix des dictionnaires va s'avérer significatif. Nous allons devoir prendre en compte l'aspect grammatical (**WordNet** et **NLTK**), tenir compte de la spécificité du langage employé sur Twitter (Prise en compte des émoticônes, de

l'argot, des mots grossiers, des fautes d'orthographe, etc.), et choisir un dictionnaire pour la polarité des mots (**SentiWordNet, Vador**).

ALGORITHMES

Pour cette partie du projet, nous devrons implémenter quatre modules :

- **L'analyse de sentiment appliqué à un tweet anglais** : Il va falloir réaliser un analyseur pour les phrases qui, à partir de la polarité des mots et de leur pondération, nous fournisse un score émotionnel (% ou indice), qui prennent en compte la spécificité de ce qu'est un tweet (court, lié à un ou plusieurs hashtags, un lieu géographique, dans un langage particulier, utilisant l'ironie ou non, etc.).

- **L'analyse d'une tendance liée à un mot-clé** : Notre programme doit pouvoir, à partir d'un mot-clé donné par l'utilisateur (terme ou hashtag en anglais), récupérer une liste de tweets du moment sur le sujet et appliquer l'analyseur de sentiment sur cette liste. Le but étant de calculer un score émotionnel pour ce mot-clé.

- **L'analyse de la Trend List** : Après avoir récupérer le top-tweet, nous aurons besoin d'un module qui réalisera l'analyse de sentiment de ces différents hashtags, dans le but de les trier du plus positif au plus négatif. Nous garderons en mémoire les tweets du hashtag le plus positif, pour les utiliser dans le module suivant.

- **Le générateur de tweet** : Après avoir déterminé la tendance du moment la plus positif, le générateur devra créer un tweet positif sur le sujet, puis le mettre en ligne. Nous ne sommes pas réellement satisfaits des informations que nous avons pu trouver sur ce sujet, nous comptons donc approfondir nos recherches davantage. A priori, nous utiliserons les chaînes de Markov ou bien des techniques de Machine Learning.

FONCTIONNALITES ET INTERFACE UTILISATEUR

Le programme comportera deux parties :

- **L'analyseur de tendance** : A partir d'un mot-clé en anglais donné par l'utilisateur, il renverra un score émotionnel (% ou indice) lié à l'analyse des tweets du moment. Si nous avons le temps, nous réaliserons l'affichage d'un nuage de points sur un graphique pour représenter visuellement le résultat.

- **Le bot « suiveur et optimiste »** : Cette partie de notre programme exploitera l'intégralité des fonctions implémentées. L'application devra réaliser, à la demande, l'analyse de la Trend List Twitter du moment pour déterminer le sujet populaire le plus positif, et ensuite générer et publier un tweet positif sur ce sujet. Le déroulement des différentes étapes sera affiché au fur et à mesure sur la sortie.

IMPLEMENTATION DU PROTOTYPE

L'implémentation s'est déroulé comme prévu, en respectant le cahier des charges. L'application a bien été réalisé en Python et en langue anglaise. Les objectifs fixés ont été atteint, l'application créée étant capable de réaliser les tâches demandées. Sera décrit ici les différentes étapes de l'implémentation, les problèmes rencontrés, et les résultats obtenus.

Pour mener à bien ce projet, nous avons décidé de séparer les différentes tâches à réaliser en différents modules, de telle sorte à ce qu'ils puissent être le plus autonome possible et éventuellement réutilisables dans d'autres contextes.

INTERACTIONS AVEC TWITTER

Le premier module réalisé a été celui permettant de communiquer avec Twitter. Il a fallu se familiariser avec l'API, ce qui ne fût pas chose facile : En effet, il existe notamment de nombreuses limitations visant à éviter les utilisations abusives (trop de requêtes/min, etc.) et d'autre part, Twitter fait bien entendu tout pour éviter le pillage de ses données par n'importe qui : Un utilisateur standard ne peut accéder qu'à une sélection spécifique de tweets, plutôt récents (la semaine passée), et sous certaines conditions. Pour avoir un accès plus complet, il faut sortir le portefeuille, cela donne donc lieu à tout un ensemble de stratégies pour réussir à faire ce que l'on souhaite sans atteindre les limites imposées. La documentation disponible sur leur site n'aide pas, elle reste vague sur beaucoup de points, il s'avère donc souvent nécessaire de mener de nombreux essais pour deviner le fonctionnement précis de l'API.

Nous avons fait le choix d'utiliser le module Tweepy, qui simplifie l'accès à Twitter. Après avoir créé un compte Twitter dédié et généré une clé API, il a été assez simple de se connecter via Tweepy et d'essayer les différentes fonctions et méthodes disponibles, mais nous avons très vite été confronté aux limitations de l'API. En effet, lorsqu'on souhaite récupérer des dizaines de milliers de tweets, selon les méthodes employées, il est facile d'atteindre très rapidement le nombre limite de requêtes par tranche de 15min pour un utilisateur. Quelques recherches sur Google suffisent à comprendre que c'est un problème récurrent pour tous les développeurs qui souhaitent faire du data mining sur Twitter.

La solution a été d'exploiter les 2 profils d'authentification possibles lorsqu'on utilise l'API : **Le profil User et le profil Application-only**. Le profil User est celui que nous utilisions déjà auparavant, il permet d'envoyer 180 requêtes par tranche de 15 minutes ainsi que d'accéder

aux commandes de publication par exemple. Pour la récupération de tweets à partir d'un mot-clé, il est possible d'obtenir 100 tweets par requête, ce qui nous donne un maximum de 18000 tweets par tranche de 15 minutes. Lors des tests, cette limite était très rapidement atteinte, il n'était pas possible de lancer plusieurs tests à la suite. Nous avons donc décider d'exploiter l'autre profil, l'authentification en tant qu'Application-only, qui possède des limites plus élevées : Avec ce profil il est possible d'émettre 450 requêtes par fenêtre de 15 minutes, ce qui nous permet de récupérer jusqu'à 45000 tweets, plus du double du profil User. L'inconvénient de ce profil est qu'il ne donne pas accès aux commandes utilisateur, telles que la publication ou autre.

Nous avons donc conçu une fonction nous permettant de passer d'un profil à l'autre selon l'action qu'on souhaite réaliser : Pour les actions de récupération de listes de tweets ou de tendances, l'authentification se fait en tant qu'Application-only, et pour les actions de publication, en tant qu'User. Les problèmes liés aux limitations ont disparu après cette modification.

Concernant la récupération de tweets, nous avons dû préciser de nombreux paramètres importants dans les requêtes, car les tweets récupérés sont par défaut tronqués, dans n'importe quelle langue, ils contiennent les retweets, et ils peuvent dater de plusieurs jours.

Concernant la récupération de la trend list, il a fallu insérer dans la requête un numéro **WOEID** (Where On Earth IDentifier) qui permet de spécifier l'emplacement géographique des tweets voulus. Nous avons choisi celui des USA, étant donné que nous voulions des tweets en anglais issus d'un pays avec une forte communauté anglophone susceptible de générer des tendances intéressantes pour nous et compréhensibles par nous (Selon le pays il peut s'avérer difficile de comprendre à quoi correspond précisément une tendance, et donc d'évaluer si le tweet généré par notre programme est pertinent ou non).

Notre module d'interaction avec Twitter contient donc 6 fonctions importantes :

- **La fonction d'authentification**, qui permet de changer entre les deux profils.
- **La fonction de récupération de tweets** qui, à partir d'un mot-clé et d'un nombre de tweets demandés, renvoie une liste de tweets.
- **La fonction de récupération de tendances** qui, à partir d'un WOEID, renvoie la trend list associée.
- **Les fonctions de publication** : pour publier un tweet, voir le dernier tweet publié, et supprimer le dernier tweet publié.

L'ANALYSEUR DE SENTIMENT

La conception de ce module a été longue à réaliser, car le fait que nous analysions des tweets rajoute une difficulté supplémentaire : En effet, il faut pouvoir traiter du texte simple, mais également du texte contenant des mots d'argot et des émoticônes.

Avant de détailler son fonctionnement, observons la structure de notre analyseur :

- Il prend en entrée du texte, qu'il normalise immédiatement.
- Le texte obtenu est tokenisé puis pos-taggé.
- On enlève les stopwords.
- Enfin, on réalise l'analyse, qui sera décrite plus bas.

La normalisation consiste ici à mettre le texte en minuscule, à remplacer tous les espaces blancs (\n\t\r…) par de simples espaces et à supprimer les URL grâce à l'utilisation des expressions régulières.

Une fois le texte nettoyé, la création d'un **tokenizer** approprié a été entrepris, avant de nous rendre compte que NLTK en fournit un adapté aux tweets, que nous avons donc décidé d'adopter. Le rôle du tokenizer est de découper la phrase en tokens, c'est-à-dire en briques de mots. L'avantage de ce tokenizer est notamment qu'il reconnait les émoticônes sans difficulté.

Ensuite, il a fallu tagger chaque token selon sa position dans la phrase. Cela permet de mieux connaitre le rôle de chaque mot dans une phrase, et ainsi de lever certaines ambiguïtés sémantiques et syntaxiques. Nous avons envisagé de réaliser nous-même ce **pos-tagger**, mais devant l'ampleur de cette tâche et la longueur de notre projet, nous avons choisi d'utiliser la fonction présente dans le module NLTK, qui fournit des résultats satisfaisants.

L'étape suivante, une fois la phrase tokenisée et les tokens taggés, consiste à enlever des **stopwords**, c'est-à-dire les mots qui n'ont aucune utilité dans l'analyse tels que « the », « of », ou bien « that ». Pour ce faire, nous avons créé une liste de mots interdits, à partir de différentes listes existantes.

Une fois le texte source préparé, nous pouvons passer à l'analyse en elle-même : Il faut simplement déterminer pour chaque token une polarité. Celle-ci sera ajouté à un total qui, selon sa valeur finale, déterminera la polarité de la phrase ou du tweet étudié. Pour déterminer la polarité d'un mot, nous avons procédé ainsi :

- Avant tout, nous convertissons le tag du token en tag correspondant à **WordNet**, par le biais d'une fonction. En effet, le tag que nous avons grâce à notre tagger ne correspond pas à celui de WordNet, la conversion est donc absolument nécessaire.

- On recherche le token taggé dans le corpus **SentiWordNet**, par le biais d'une fonction de NLTK. Si le token est présent, on récupère sa polarité. En réalité, on ne récupère pas directement cette donnée, mais un synset, c'est-à-dire une structure correspondante à un groupe synonymique. Ce synset possède un score positif, négatif et objectif. Ces scores sont tous une note comprise entre 0 et 1. Nous n'avons pas utilisé le score objectif dans l'analyse, seulement la combinaison des scores positifs et négatifs, qui constitue la polarité.

- Si le token est introuvable, on le cherche dans le corpus **Vader**, par le biais d'une fonction de NLTK, et on ajoute sa polarité s'il est présent. Vader nous fournit, en plus des mots classiques, des polarités pour les émoticônes et aussi certains mots d'argots.

  Avant de l'utiliser, nous avons essayé de trouver des corpus pour obtenir la polarité des émoticônes, ils en existent quelques-uns que nous aurions pu utiliser. Mais c'est le problème de la polarité des mots d'argots qui nous a fait adopter Vader. En effet, aucun corpus ne réglait véritablement le problème. Nous avons tenté de déterminer la polarité des mots d'argots pendant un moment en concevant un module qui récupérait leurs définitions sur le site UrbanDictionary et en faisait l'analyse de sentiments. Sans même évoquer l'aspect récursif du problème, beaucoup de définitions sont plus humoristiques que sérieuses, elles produisaient de ce fait des scores de polarité qui n'ont aucun sens.

  Nous avons donc utilisé les polarités données par Vader, plus précisément le « compound », qui est la combinaison du score positif et négatif d'un mot. Le score obtenu est un nombre compris entre -1 et 1.

- Si on n'a pas pu obtenir sa polarité, ni avec SentiWordNet, ni avec Vader, on ignore simplement le mot.

- Pour affiner l'analyse, nous avons rajouté une **prise en compte de la négation**, susceptible d'inverser considérablement la polarité des phrases. Pour ce faire, avant même la détermination de la polarité, l'analyseur regarde si le token courant est présent dans la liste des mots négatifs (une liste ressemblant à celle des stopwords, mais composée uniquement de mots susceptibles d'inverser les polarités, que nous avons construite nous-même). S'il n'est pas dans la liste, on détermine sa polarité comme expliqué précédemment, sinon on garde en mémoire qu'il faudra inverser la polarité du mot suivant. Nous avons mené plusieurs essais pour savoir si le mieux était d'inverser la polarité d'un, deux ou trois mots suivants. Le module d'évaluation de l'analyseur (voir plus bas) nous a fourni des résultats qui montraient qu'inverser uniquement un seul mot suivant était le plus efficace.

Une fois les polarités déterminées, il suffit d'en faire le total, et de le pondérer par le nombre de mots significatifs (c'est-à-dire polarisés) de la phrase. On obtient ainsi la polarité de la phrase source, sous la forme d'un score compris entre -1 et 1.

```
even = <even.r.01: PosScore=0.125 NegScore=0.0> <ObjScore=0.875>
i'm = <sentivader: PosScore=0.0 NegScore=0.0 NeuScore=1.0 Compound=0.0>
angry = <angry.a.01: PosScore=0.375 NegScore=0.375> <ObjScore=0.25>
never = <inverter>
kill = <kill.v.01: PosScore=0.0 NegScore=0.5> <ObjScore=0.5>
someone = <person.n.01: PosScore=0.0 NegScore=0.0> <ObjScore=1.0>

even when i'm angry, i could never kill someone :
pos : 0.33333 | neg : 0.125 | obj : 0.54167 | polarity : 0.20833
```

Exemple avec la phrase "Even when i'm angry, i could never kill someone" correctement évaluée comme positive.

L'EVALUATION DE L'ANALYSEUR DE SENTIMENT

Pour évaluer sa fiabilité, nous avons conçu un **module d'évaluation** qui utilise notre analyseur de sentiment comme classifieur. A partir d'un corpus de textes déjà triés comme positifs ou négatifs que notre analyseur va reclasser, on peut déterminer des scores tels que la F-mesure.

Nous avons utilisé 4 corpus différents, dont voici les résultats :

```
Twitter Samples : (corpus de NLTK, 10.000 tweets)
F-mesure Pos : 0.8970753655793026
F-mesure Neg : 0.8823403343334761
Correct/Total : 0.8902 | Incorrect/Total : 0.1098

Movie Reviews : (corpus de NLTK, 2.000 critiques)
F-mesure Pos : 0.6938610662358643
F-mesure Neg : 0.5026246719160105
Correct/Total : 0.621 | Incorrect/Total : 0.379

Sentiment140 (panel test, 500 tweets) :
F-mesure Pos : 0.7817745803357316
F-mesure Neg : 0.6976744186046511
Correct/Total : 0.7465181058495822 | Incorrect/Total : 0.25348189415041783

Sentiment140 (panel train, 1.600.000 tweets) :
F-mesure Pos : 0.6936528504043472
F-mesure Neg : 0.5205079080058037
Correct/Total : 0.626155 | Incorrect/Total : 0.373845
```

Les résultats sont variables selon les corpus bien entendus, mais semblent plutôt satisfaisants à la vue des techniques utilisées.

A titre informatif, voici les résultats F-mesure d'autres analyseurs :

- **SentiWordNet** : Environ 70%
- **Vader** : Environ 74%
- Classifieurs utilisant le **Machine Learning** : Peut dépasser 90%

PREPARATION DE LA GENERATION DE TEXTES

Pour préparer la partie suivante, nous avons créé un ensemble de modules destinés à générer des n-grams depuis nombreuses sources. En voici la liste et l'utilité :

- **Un module de conversion .epub** -> .txt pour pouvoir utiliser des ebooks. Ce module avait été réalisé par l'un de nous pour un précédent projet (un prédicteur de texte)

- **Un module de conversion .json** -> .txt pour pouvoir utiliser des données issues de Reddit. En effet, les commentaires de Reddit sont un bon exemple de langage courant, c'est donc une ressource intéressante pour notre projet.

- **2 modules « lexique » et « n-grams »** : Ils permettent, à partir de sources .txt dans un répertoire précis, de générer un lexique et des listes de 2-grams, 3-grams et 4-grams de mots suivants. Les sources peuvent être des ebooks, .epub ou des fichiers .json de Reddit ayant été convertis en .txt. Une fois ces listes générées, elles peuvent être filtrées selon des nombres minimum d'occurrence, et sauvegardées dans un répertoire prévu à cet effet. Il serait long et peu utile de détailler le fonctionnement précis de ces modules (**Le projet complet est disponible sur Dropbox**). Il est également possible de générer les « contexte-grams », c'est-à-dire les 2-grams d'apparition conjointe de 2 mots différents dans un texte.

  Pour simplifier la génération de n-grams, **une fonction réalisant toutes ces opérations** a été écrite : elle se charge de lire les .txt dans le répertoire prévu, de construire le lexique et les n-grams, de les filtrer selon les critères donnés en argument, puis de les sauvegarder.

  Une autre fonction a été réalisée, pour pouvoir **générer ces listes de n-grams « à la volée »** (en les conservant en mémoire sans les sauvegarder sur le disque). Elle peut, à partir d'un texte ou d'une liste de phrases donnés en argument, faire la même chose que la fonction précédente. Elle est utile si on souhaite générer les n-grams d'une liste de tweets par exemple…

Les modules précédents ayant été préparés, il est désormais possible réaliser ce module de génération, ayant pour but de créer du texte à partir de n-grams. Cela est possible aussi bien **à partir de n-grams sauvegardés (donc générés à partir de ebooks ou de Reddit) qu'à partir de n-grams construits « à la volée » (avec des tweets par exemple).**

Pour ce faire, il suffit de créer une phrase ne contenant que le mot-commençant « <s> », puis de piocher des n-grams jusqu'à tomber sur un mot-terminant tel que « </s> » ou « </!> ». Le processus de pioche est relativement simple et ordonné : il regarde si des 4-grams commençant par les 3 derniers mots existent, sinon il regarde si des 3-grams commençant par les 2 derniers mots existent, sinon il regarde si des 2-grams commençant par le dernier mot existent, sinon il renvoie une erreur.

La pioche favorise donc les n-grams au n le plus grand en premier, ce qui a pour particularité de générer des phrases plus sensées. Quand on a pioché les n-grams, on ne garde que ceux qui ont la plus grande occurrence, et on choisit parmi eux au hasard celui qui déterminera le mot suivant.

Ce simple procédé permet de générer des phrases à partir d'un corpus de n-grams.

Le temps nous a manqué, mais nous aurions voulu **mélanger les corpus**, c'est-à-dire le corpus des n-grams générés à partir des ebooks et de Reddit, avec le corpus des n-grams générés à la volée à partir de tweets. Ce n'aurait pas été un simple mélange, il aurait fallu concevoir un algorithme capable de créer un mélange intelligent entre les deux, dans le but de créer des tweets nouveaux et cohérents.

Le problème que nous avons eu ici était que les tweets que nous avons générés étaient rarement nouveaux, car le nombre de tweets utilisés lors de la génération des n-grams à la volée n'était pas suffisants (seulement 10.000). Le programme avait donc une grande probabilité de regénérer des tweets déjà existants.

D'autre part, lorsqu'il s'agissait de générer du texte à partir de l'autre corpus (ebooks/Reddit), le résultat était parfois insensé mais il avait le mérite d'être assurément original.

C'est pour cette raison que nous aurions voulu créer un algorithme capable de mélanger ces deux corpus pour rendre plus original les tweets que nous aurions générés.

Ce module est le dernier de notre projet, il utilise l'intégralité des modules précédents. Celui-ci contient plusieurs fonctions décrites ici, qui réalisent les objectifs du projet :

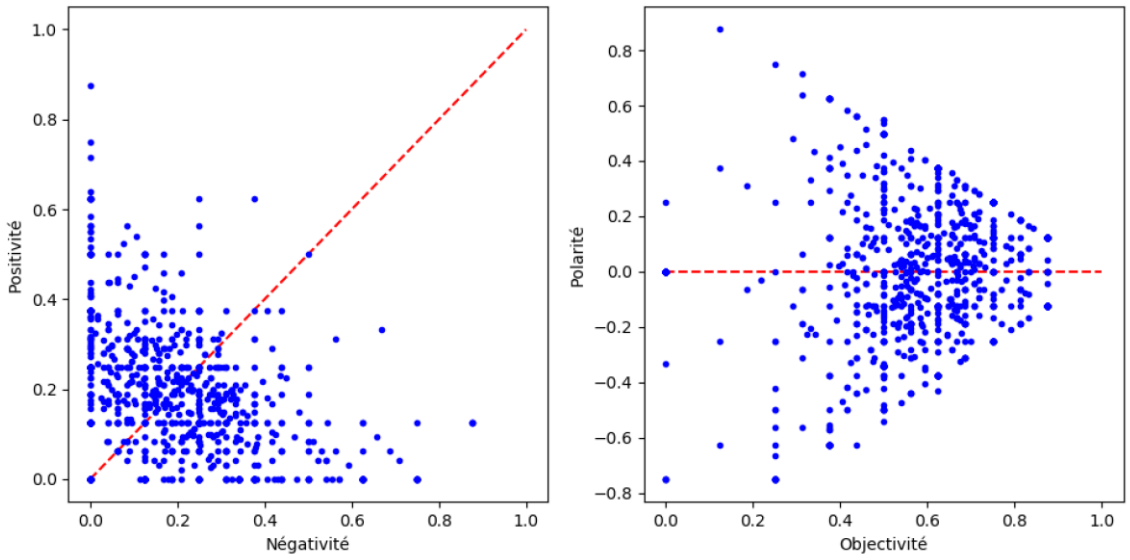
Evaluation d'une tendance :

A partir d'une tendance (n'importe quel mot-clé), cette fonction récupère un nombre de tweets (ce nombre est indiqué en argument), conduit une analyse de sentiment pour déterminer leurs polarités, puis en fait la moyenne. Cette moyenne constitue le score pour le mot-clé donné. Nous avons intégré le module **MatPlotLib**, qui permet de visualiser les résultats.

Exemple pour le mot-clé « trump » :

RESULTAT :

```
trump : 0.024
```



On a un score de tendance très légèrement positif qui traduit probablement une division dans l'opinion. On observe également bien cette lutte sur le graphique, où les points (c'est-à-dire les tweets) sont répartis assez équitablement de chaque côté de la ligne.

Evaluation des tendances aux USA :

Cette fonction récupère la trend list des USA, analyse chaque tendance, et crée un top de la plus positive à la plus négative. Voici un exemple, réalisé à partir des 10 tendances les plus populaires du moment, avec 100 tweets par tendance :

```
Analyse des tendances... [10]x[100]

* TOP 10 POSITIVE TRENDS USA *
#mobileliving : 0.7
#FridayFeeling : 0.46
#FunFactFriday : 0.33
#gayface : 0.28
#askghost : 0.25
#archivesanimals : 0.19
#NRAConvention : 0.18
#Trumpcare : 0.17
#NickiDay2 : 0.16
#WhatStarWarsTaughtMe : 0.09
```

Création de contexte :

Cette fonction permet, à partir d'un mot clé, de construire son contexte, c'est-à-dire une liste de mots qui lui est généralement associé. Pour ce faire, la fonction récupère simplement les tweets où le mot-clé apparait, puis construit les « contexte-grams », qui sont les 2-grams d'apparition conjointe de 2 mots différents dans ces tweets. Il génère ensuite une liste des mots apparaissant le plus avec notre mot-clé. Voici un exemple avec 5000 tweets et le mot-clé « Trump » :

```
Trend : trump
Récupération de tweets de la tendance... [5000]
president
donald
nra
like
says
administration
mueller
giuliani
people
@realdonaldtrump
judge
000
one
stormy
news
speech
kanye
lies
```

Génération d'un tweet positif à propos de la tendance la plus populaire et positive :

Cette fonction est l'aboutissement de notre projet. Après avoir déterminé la tendance la plus populaire et positive aux USA, elle récupère des tweets à propos de cette tendance, puis génère des n-grams à la volée (à partir des tweets récupérés).

Une fois les n-grams créés, un certain nombre de phrases sont générées puis analysées avec notre analyseur de sentiment, dans le but de garder la plus positive, qui sera publiée.

Voici un exemple :

```
Analyse des tendances... [5]x[500]
Meilleure Tendance : #c2nextgen : 0.272
Récupération de tweets de la tendance... [5000]
Génération de tweets... [500]
Dictionnaires générés.
Analyse des tweets générés...
Meilleur Tweet Généré : if we have diversity at the table it makes for a better table  #c2nextgen : 0.875
"If we have diversity at the table it makes for a better table  #c2nextgen" : Publié !
```

Avec sa publication automatisée :

CONCLUSION DU PROJET

LES AMELIORATIONS

Il y a 3 choses qui nous aurions voulu améliorer, dans l'ordre :

- **La génération de tweets** : Comme expliqué plus haut, il arrive souvent que le tweet généré existe déjà. En effet, ayant un panel de tweets trop restreint, les n-grams ont une grande chance de produire que des tweets déjà existants. La solution que nous avions envisagée était de mélanger les n-grams générés par les tweets, avec des n-grams générés à partir de Ebooks ou de Reddit (que nous avons généré auparavant). Il s'agissait de créer un algorithme permettant de rendre plus originale la génération des tweets, en ajoutant par touche des n-grams extérieurs à ceux présents dans les tweets. Le manque de temps nous a empêché de réaliser cette tâche, qui nous semblait la plus importante.

- **L'analyseur de sentiment** : Il est toujours possible d'améliorer l'analyseur, de le perfectionner. Le résultat actuel est le fruit d'un long travail, principalement de familiarisation avec ces techniques. Cette expérience donne aussi envie d'en réaliser un avec des techniques de Machine Learning, qui semblent assez prometteuses.

- **Une interface graphique** : Nous aurions aimé aller jusqu'au bout, en réalisant une interface graphique très légère, où l'utilisateur n'aurait plus qu'à cliquer sur un bouton qui provoquerait la génération d'un tweet. Cette amélioration est de loin la plus dispensable.

BILAN

Nous sommes relativement satisfaits du résultat, étant donné l'ampleur de l'objectif de départ. L'analyseur n'est pas parfait, le générateur de tweets non plus, mais le fait est que notre programme réalise bien ce que nous voulons réaliser. Il est amusant de constater que le bot, malgré qu'il soit un bot, se fait parfois « liker » ou « retweeter ». Il arrive parfois que nous ne comprenions même pas le sujet de la tendance à propos de laquelle le bot vient de tweeter. C'était un projet intéressant, sur lequel nous aurons plaisir à revenir pour, pourquoi pas, l'améliorer dans le futur.

Lien twitter du Bot : https://twitter.com/MoutonBot

Lien du projet : https://www.dropbox.com/s/6xp2nvlvqunaa2k/TwittoBot.zip?dl=0