



Ecole Nationale Supérieure de l'Informatique et de  
l'Analyse des Systèmes

2014/2015

# Rapport du Projet d'Intelligence Artificielle Opinion Mining

---

Encadré par :

*Prof Mme. BENBRAHIM Houda*

*Prof Mme. BEKKALI Hanane*

*Prof M. KASSOU Ismaïl*

Réalisé Par :

*MAIOUAK Mariem*

*G6*

*MESSOUDI Soundouss*

*G6*

---

## Table des matières

Présentation du projet .....	2
1. Introduction à l'Opinion Mining [1].....	2
2. Description du projet.....	2
Explication du code .....	4
1. Prétraitement .....	4
2. Postagging .....	6
3. Calcul des scores.....	7
Interface graphique .....	11
1. L'API CAPI [3] .....	11
2. Interface de prétraitement.....	11
3. Interface de Post-Tagging.....	11
Webographie .....	13

# Présentation du projet

## 1. Introduction à l'Opinion Mining [1]

### \* Définition

En informatique, l'Opinion Mining (aussi appelé Sentiment Analysis) est l'analyse des sentiments à partir de sources textuelles dématérialisées sur de grandes quantités de données (Big Data).

Ce procédé apparaît au début des années 2000 et connaît un succès grandissant dû à l'abondance de données provenant de réseaux sociaux, notamment celles fournies par Twitter.

L'objectif de l'Opinion Mining est d'analyser une grande quantité de données afin d'en déduire les différents sentiments qui y sont exprimés. Les sentiments extraits peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté.

### \* Analyse de données

Le but de l'analyse de données est de déterminer si le sentiment dégagé par une phrase est positif ou négatif. La principale difficulté de l'analyse réside au cœur même de l'utilisation de la langue. Le sentiment dégagé par une phrase dépend directement du contexte dans laquelle elle est utilisée, du type de langage, ainsi que de la personne qui l'a écrite... En réalité, il existe une multitude de facteurs de plus ou moins grande influence qui altère le sentiment suscité par un propos.

Il existe des outils permettant d'identifier le sentiment dégagé par un texte. Parmi les outils les plus connus on trouve SentiWordNet qui est une extension du WordNet (permet de savoir à l'aide de groupe de synonymes si un mot est positif ou non). Il attribue à chaque groupe de synonymes provenant de WordNet, trois scores de sentiment : la positivité, la négativité, l'objectivité. SentiWordNet est justement l'outil qui va être utilisé dans ce projet.

## 2. Description du projet

De nos jours, les sites web du commerce électronique et les media sociaux jouent un rôle de plus en plus important pour la commercialisation des produits. Ainsi, ces sites permettent aux consommateurs de donner leurs avis sur les produits commercialisés. Ces opinions, souvent écrites sous forme de commentaires, sont analysées par les entreprises à des fins très différentes parmi lesquelles la comparaison des opinions concernant leurs produits avec celles de leurs concurrents. Pour cela, une analyse automatique s'impose grâce aux techniques d'« Opinion Mining ».

Le projet d'IA de cette année consiste à écrire un programme en LISP qui va analyser les opinions (commentaires) des clients d'un produit spécifique et donner le pourcentage de ceux qui sont satisfaits (positif), non satisfaits (négatif) et neutres.

Le programme doit répondre aux critères suivants :

1. Prendre en entrée un fichier texte qui contient les commentaires (un commentaire par ligne) concernant un produit donné.
2. Faire un prétraitement à ces commentaires (éliminer les ponctuations, les 'stop words').
3. Appliquer le « postagging » pour déterminer les noms, les verbes et les adjectifs.
4. Appliquer « SentiWordNet » pour les adjectifs présents dans un commentaire donné pour avoir la polarité et le degré de la polarité de ces adjectifs.
5. Prendre en considération les cas de négation : « not happy » (CRITERE TRES IMPORTANT)
6. Déduire l'opinion d'un commentaire (satisfait, non satisfait, neutre)
7. Faire un comptage et déduire les pourcentages de satisfaction, non satisfaction et de neutralité concernant un produit donné.
8. Afficher le résultat convenablement (interface, diagramme,...)

## Explication du code

### 1. Prétraitement

L'objectif de cette première étape est de :

1. Prendre en entrée un fichier texte qui contient les commentaires (un commentaire par ligne) concernant un produit donné.
2. Faire un prétraitement à ces commentaires (éliminer les ponctuations, les 'stop words').

Tout d'abord, nous avons le fichier d'entrée du produit sous format .txt avec un commentaire par ligne.

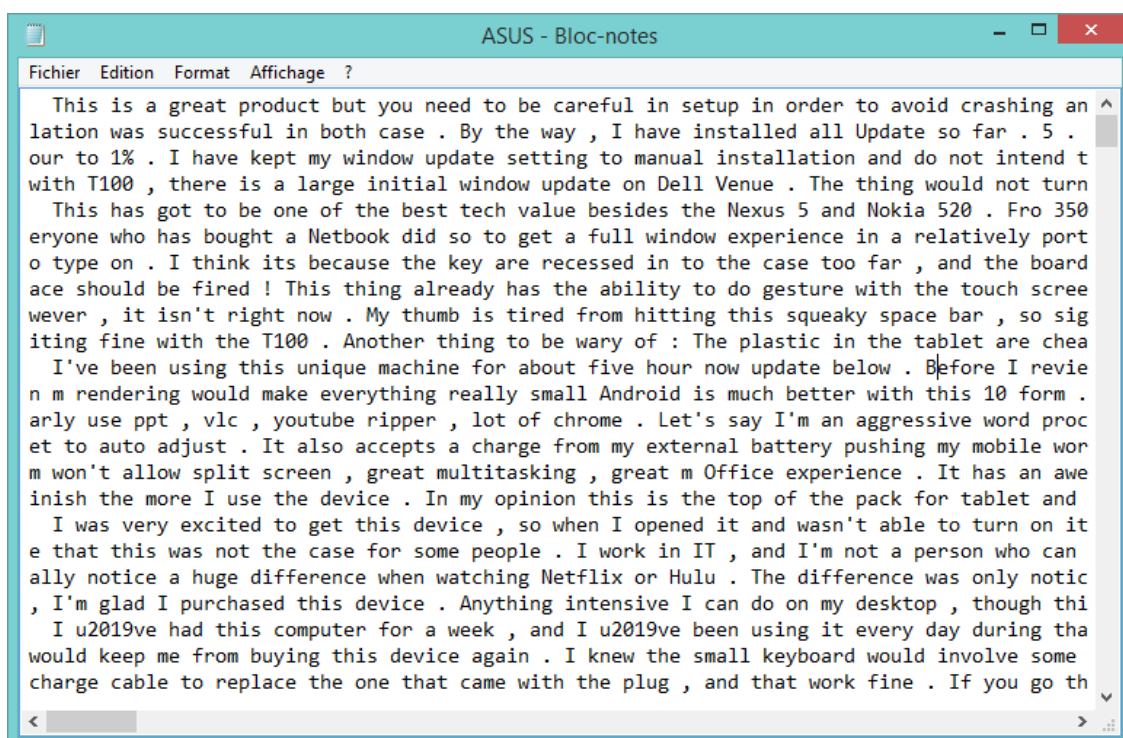
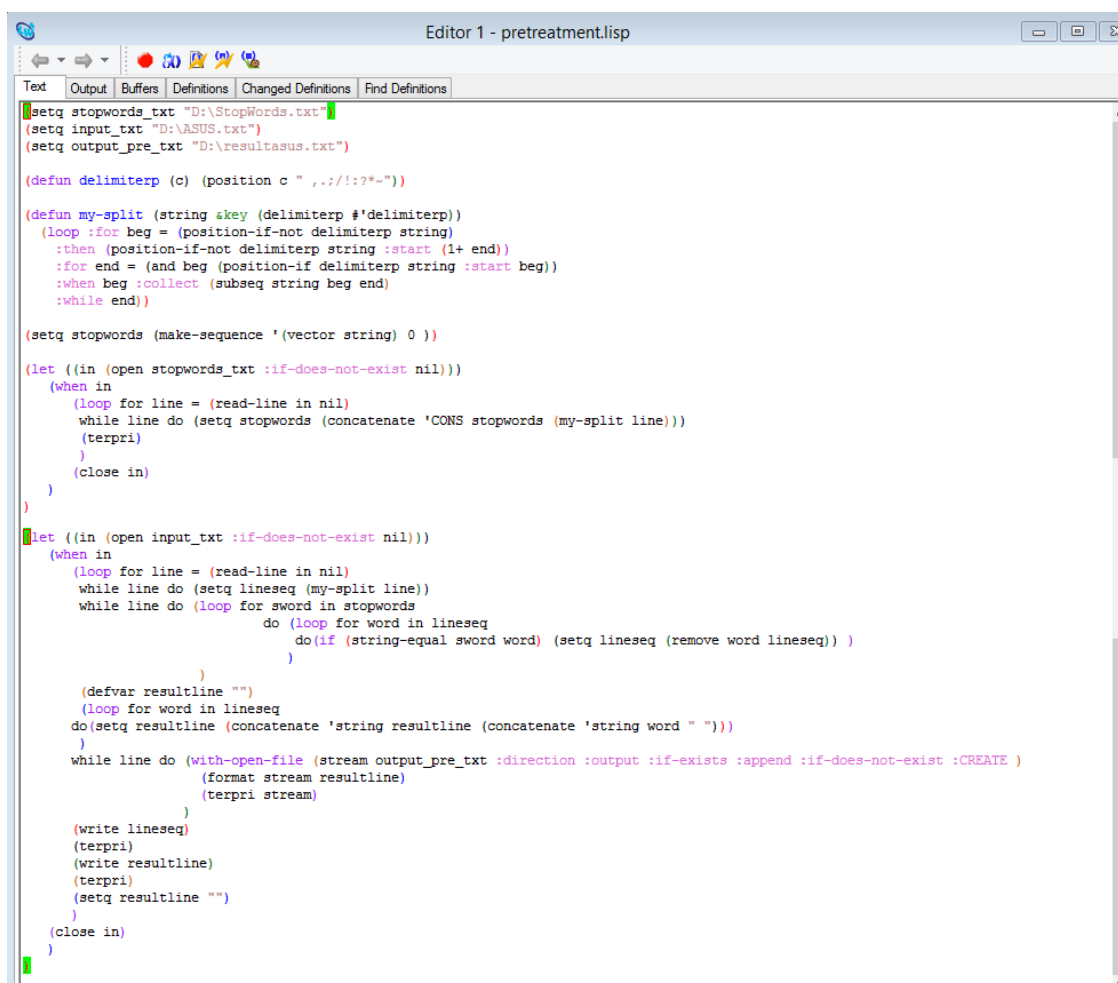


Figure 1: Fichier d'entrée du produit ASUS.

Nous avons donné les chemins des fichiers «ASUS.txt», «StopWords.txt» et «resultasus.txt» respectivement comme fichier d'entrée des commentaires, fichier d'entrée des StopWords et fichier de sortie du résultat de prétraitement. Après nous avons utilisé la fonction «my-split» retrouvée sur le site « Stackoverflow.com » pour transformer une chaîne de caractères en une liste dont le séparateur est indiqué par «delimiterp». [2] Cette fonction a été utilisée sur les deux fichiers de commentaires «ASUS.txt» et de stopwords «StopWords.txt» pour pouvoir comparer les deux, et enregistrer les mots qui ne figurent pas parmi les stopwords dans un fichier de sortie «resultasus.txt».

Il faut noter que le fichier «StopWords.txt» donné a été changé en en supprimant le mot «not» pour pouvoir traiter son cas pour la question 5 du projet.

Voici le code lisp du prétraitement :



```

Editor 1 - pretreatment.lisp

Text Output Buffers Definitions Changed Definitions Find Definitions

(setq stopwords_txt "D:\\StopWords.txt")
(setq input_txt "D:\\ASUS.txt")
(setq output_pre_txt "D:\\resultasus.txt")

(defun delimiterp (c) (position c " ,./!?:*~"))

(defun my-split (string &key (delimiterp #'delimiterp))
  (loop :for beg = (position-if-not delimiterp string)
        :then (position-if-not delimiterp string :start (1+ end))
        :for end = (and beg (position-if delimiterp string :start beg))
        :when beg :collect (subseq string beg end)
        :while end))

(setq stopwords (make-sequence '(vector string) 0))

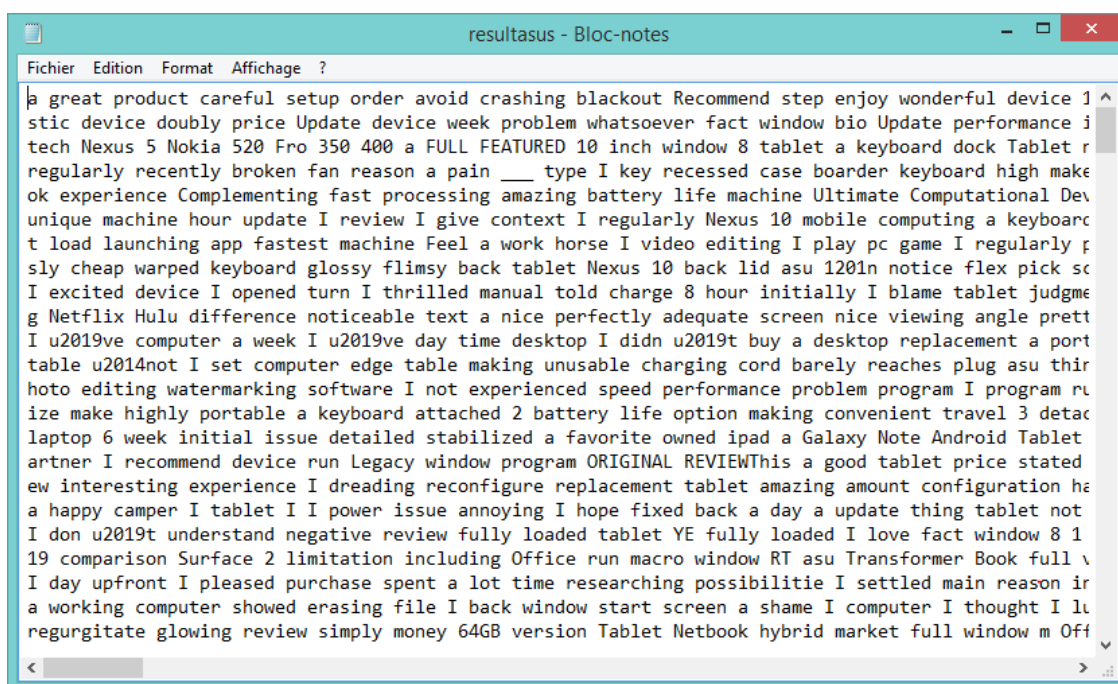
(let ((in (open stopwords_txt :if-does-not-exist nil)))
  (when in
    (loop for line = (read-line in nil)
          while line do (setq stopwords (concatenate 'CONS stopwords (my-split line)))
          (terpri)
          )
    (close in)
  )
)

(let ((in (open input_txt :if-does-not-exist nil)))
  (when in
    (loop for line = (read-line in nil)
          while line do (setq lineseq (my-split line))
          while line do (loop for word in lineseq
                              do (loop for sword in stopwords
                                      do (if (string-equal sword word) (setq lineseq (remove word lineseq)) )
                              )
          )
    (defvar resultline "")
    (loop for word in lineseq
          do (setq resultline (concatenate 'string resultline (concatenate 'string word " ")))
          )
    while line do (with-open-file (stream output_pre_txt :direction :output :if-exists :append :if-does-not-exist :CREATE)
                    (format stream resultline)
                    (terpri stream)
                    )
    (write lineseq)
    (terpri)
    (write resultline)
    (terpri)
    (setq resultline "")
    )
  (close in)
  )
)

```

Figure 2: Code Lisp de Prétraitement

Le résultat de l'exécution de ce code est la création du fichier suivant :



```

resultasus - Bloc-notes

Fichier Edition Format Affichage ?

a great product careful setup order avoid crashing blackout Recommend step enjoy wonderful device 1
stic device doubly price Update device week problem whatsoever fact window bio Update performance i
tech Nexus 5 Nokia 520 Fro 350 400 a FULL FEATURED 10 inch window 8 tablet a keyboard dock Tablet r
regularly recently broken fan reason a pain ___ type I key recessed case boarder keyboard high make
ok experience Complementing fast processing amazing battery life machine Ultimate Computational Dev
unique machine hour update I review I give context I regularly Nexus 10 mobile computing a keyboarc
t load launching app fastest machine Feel a work horse I video editing I play pc game I regularly p
sly cheap warped keyboard glossy flimsy back tablet Nexus 10 back lid asu 1201n notice flex pick sc
I excited device I opened turn I thrilled manual told charge 8 hour initially I blame tablet judgme
g Netflix Hulu difference noticeable text a nice perfectly adequate screen nice viewing angle prett
I u2019ve computer a week I u2019ve day time desktop I didn u2019t buy a desktop replacement a port
table u2014not I set computer edge table making unusable charging cord barely reaches plug asu thir
hoto editing watermarking software I not experienced speed performance problem program I program ru
ize make highly portable a keyboard attached 2 battery life option making convenient travel 3 deta
laptop 6 week initial issue detailed stabilized a favorite owned ipad a Galaxy Note Android Tablet
arther I recommend device run Legacy window program ORIGINAL REVIEWThis a good tablet price stated
ew interesting experience I dreading reconfigure replacement tablet amazing amount configuration ha
a happy camper I tablet I I power issue annoying I hope fixed back a day a update thing tablet not
I don u2019t understand negative review fully loaded tablet YE fully loaded I love fact window 8 1
19 comparison Surface 2 limitation including Office run macro window RT asu Transformer Book full v
I day upfront I pleased purchase spent a lot time researching possibilitie I settled main reason ir
a working computer showed erasing file I back window start screen a shame I computer I thought I lu
regurgitate glowing review simply money 64GB version Tablet Netbook hybrid market full window m Off

```

Figure 3: Résultat du prétraitement "resultasus.txt"



### 3. Calcul des scores

4. Appliquer « SentiWordNet » pour les adjectifs présents dans un commentaire donné pour avoir la polarité et le degré de la polarité de ces adjectifs.

6. Dédurre l'opinion d'un commentaire (satisfait, non satisfait, neutre)

Au début, nous avons extrait à partir du fichier des mots postagés «asuspostagged.txt» les adjectifs seulement. Ceci est possible en sélectionnant uniquement les mots identifiés par «JJ». Le fichier de sortie est le suivant :





Figure 6: Extrait des adjectifs "resultadjasus.txt"

Pour cette partie, nous aurons besoin de 3 autres types de la fonction « my-split » avec des séparateurs différents : une avec la séparatrice tabulation, une autre avec le séparateur espace et la dernière avec le séparateur #. Ces fonctions nous serviront pour extraire les données dont nous avons besoin pour calculer le score depuis une version modifiée du fichier «SentiWordNet» qui ne contient pas d'entête et ne contient que les adjectifs :

1	a	00001740	0.125	0	able#1 (usually followed by 'to') having the necessary means or skill or kno
2	a	00002098	0	0.75	unable#1 (usually followed by 'to') not having the necessary means or skil
3	a	00002312	0	0	dorsal#2 abaxial#1 facing away from the axis of an organ or organism; "the abaxi
4	a	00002527	0	0	ventral#2 adaxial#1 nearest to or facing toward the axis of an organ or organism;
5	a	00002730	0	0	acroscopic#1 facing or on the side toward the apex
6	a	00002843	0	0	basiconic#1 facing or on the side toward the base
7	a	00002956	0	0	abducting#1 abducent#1 especially of muscles; drawing away from the midline of t
8	a	00003131	0	0	adductive#1 adducting#1 adducent#1 especially of muscles; bringing together or d
9	a	00003356	0	0	nascent#1 being born or beginning; "the nascent chicks"; "a nascent insurgency"
10	a	00003553	0	0	emerging#2 emergent#2 coming into existence; "an emergent republic"

Figure 7: Fichier "SentiWordNet" modifié

Il faut aussi noter que nous avons réalisé deux codes, le premier pour calculer le score positif et négatif seulement et l'autre pour calculer le score positif, négatif et neutre et ce, car quand nous introduisons le score neutre, presque tous les commentaires sont comptés comme neutres. Nous avons décidé donc d'exposer la version avec la polarité positive et négative, l'autre version peut être vérifiée à partir du code Lisp donné.

Voici donc le code Lisp de cette partie :

```

Editor 1 - tagtofinalresultu2.lisp

Text Output Buffers Definitions Changed Definitions Find Definitions

(setq POS_score (make-hash-table :test 'equal))
(setq NEG_score (make-hash-table :test 'equal))
(setq norm (make-hash-table :test 'equal))

let ((in (open SWN_txt :if-does-not-exist nil)))
  (when in
    (loop for line = (read-line in nil)
          while line do (setq lineseq (my-split1 line))
          while line do (setq words (my-split2 (elt lineseq 4)))
          while line do (loop for word in words
                             while word do (setf (gethash (elt (my-split3 word) 0) POS_score) (+ (* (/ 1 (with-input-from-string (in (
rd) 0) POS_score) (gethash (elt (my-split3 word) 0) POS_score) 0))) (with-input-from-string (in (elt lineseq 2)) (read in))) (if (gethash (elt (my-split3 wo
ut-from-string (in (elt (my-split3 word) 1)) (read in))) (with-input-from-string (in (elt lineseq 3)) (read in))) (if (gethas
h (elt (my-split3 word) 0) NEG_score) (gethash (elt (my-split3 word) 0) NEG_score) 0))) (with-input-from-string (in (elt lineseq 3)) (read in))) (if (gethas
string (in (elt (my-split3 word) 1)) (read in))) (if (gethash (elt (my-split3 word) 0) norm) (gethash (elt (my-split3 word) 0)
) norm) 0))
          )
    )
  )
  (close in)
)

(loop for key being the hash-keys of POS_score
      while key do (setf (gethash key POS_score) (/ (gethash key POS_score) (gethash key norm) ))
      while key do (setf (gethash key NEG_score) (/ (gethash key NEG_score) (gethash key norm) ))
)

(setq adjectives (make-sequence '(vector string)
                                0
                                ))
(defvar resultline "")

(setq neg_line 0)
(setq pos_line 0)

(setq c 0)

(let ((in (open input_postagged :if-does-not-exist nil)))
  (when in
    (loop for line = (read-line in nil)
          while line do (setq lineseq (my-split line))
          while line do (setq neg_words 0)
          while line do (setq pos_words 0)
          while line do (setq neu_words 0)
          while line do (setq notword nil)
          while line do (loop for word in lineseq
                             while word do (setq neg 0)
                             while word do (setq pos 0)
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (setq resultline (concatenate '
string resultline (concatenate 'string (subseq word 0 (- (length word) 3)) " ")))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethash (string-downcase (
subseq word 0 (- (length word) 3)) NEG_score) (setq pos (gethash (string-downcase (subseq word 0 (- (length word) 3)) POS
_score))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethash (string-downcase (
subseq word 0 (- (length word) 3)) NEG_score) (setq neg (gethash (string-downcase (subseq word 0 (- (length word) 3)) NEG
_score))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethas
h (string-downcase (subseq word 0 (- (length word) 3)) NEG_score) (if notword (setq c neg))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethas
h (string-downcase (subseq word 0 (- (length word) 3)) NEG_score) (if notword (setq neg pos))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethas
h (string-downcase (subseq word 0 (- (length word) 3)) NEG_score) (if notword (setq pos c))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethas
h (string-downcase (subseq word 0 (- (length word) 3)) NEG_score) (if (> pos neg) (setq pos_words (+ 1 pos_words))))
                             while word do (if (string-equal (subseq word (- (length word) 2)) "JJ") (if (gethas
h (string-downcase (subseq word 0 (- (length word) 3)) NEG_score) (if (> neg pos) (setq neg_words (+ 1 neg_words))))
                             while word do (if (string-equal word "not_RB") (setq notword t) (setq notword nil))
                             )
          while line do (write pos_words)
          (terpri)
          while line do (write neg_words)
          (terpri)
          (terpri)
          while line do (if (> pos_words neg_words) (setq pos_line (+ 1 pos_line)))
          while line do (if (< pos_words neg_words) (setq neg_line (+ 1 neg_line)))
          while line do (write pos_words)
          (terpri)
          while line do (write neg_words)
          (terpri)
          (terpri)
          while line do (if (> pos_words neg_words) (setq pos_line (+ 1 pos_line)))
          while line do (if (< pos_words neg_words) (setq neg_line (+ 1 neg_line)))
          while line do (with-open-file (stream output_adjectives :direction :output :if-exists :append :if-does-not-exist :CREA
TE)
                                (format stream resultline)
                                (terpri stream)
                                )
          while line do (setq resultline "")
          )
    )
  )
  (close in)
)

(write pos_line)
(write neg_line)

```

Figure 8: Code Lisp de calcul de polarité

A partir du SentiWordNet, nous avons construit une liste dont le séparateur est tabulation, ceci nous a permis d'identifier le 3ème élément de la liste comme le score positif du synset, le 4ème élément comme son score négatif et le 5ème élément comme les mots qui appartiennent à ce synset. Nous avons pu récupérer l'ensemble `mot#sensnumber` en

utilisant la version de « my-split » avec le séparateur espace sur le 5ème élément de la liste pour constituer une autre liste contenant mot#sensnumber puis nous avons utilisé la version my-split avec comme séparateur # pour obtenir individuellement ces deux éléments.

Ainsi, nous pouvions donc calculer pour chaque mot figurant dans le SentiWordNet son score en appliquant la fonction suivante qui calcule une moyenne pondérée du score positif :

$$\text{Score Positif (mot)} = \frac{\text{somme} [(1/\text{sense number}) * \text{score positif (synset)}]}{\text{somme} (1/\text{sense number})}$$

Ces scores ont été enregistré dans une table de hashage avec comme clé le mot et comme valeur son score. Nous avons fait de même pour la table de hashage du score négatif.

Si le mot est précédé par un « not », les scores de ce dernier sont inversés.

Enfin, il ne restait plus qu'à compter le nombre de mots positifs et de mots négatifs de tout le commentaire, comparer ces deux nombres pour déterminer si l'opinion de ce commentaire est satisfait ou non satisfait et ensuite de compter le nombre de commentaires positifs et négatifs pour avoir le pourcentage.

Voici le résultat obtenu :

```

Listener 4
Listener Output
3
5
6
1
4
2
9
4
0
1
7
2
T
CL-USER 24 > (write pos_line)
111
111
CL-USER 25 > (write neg_line)
31
31
CL-USER 26 >
Ready.

```

Figure 9: Résultat d'opinion mining pour ASUS

Pour le produit ASUS, il y'a 111 opinions satisfaites et 31 opinions non satisfaites. Donc en général, ce produit est satisfaisant.

# Interface graphique

## 1. L'API CAPI [3]

LispWorks comprend CAPI, une API multiplateforme pour le développement du programme GUI. CAPI est puissant et facile à utiliser. Pour une véritable capacité multiplateforme, il est important de rester avec 100 % -Pure CAPI.

Nous avons utilisé CAPI pour répondre à la dernière question du projet :

8. Afficher le résultat convenablement (interface, diagramme,...).

## 2. Interface de prétraitement

Pour l'interface de prétraitement, nous avons donné à l'utilisateur le choix de faire l'opinion mining sur l'un des 4 produits donnés comme data, sinon de parcourir son pc ou d'entrer directement le chemin d'accès au fichier du produit dont il souhaite faire l'opinion mining comme suit :

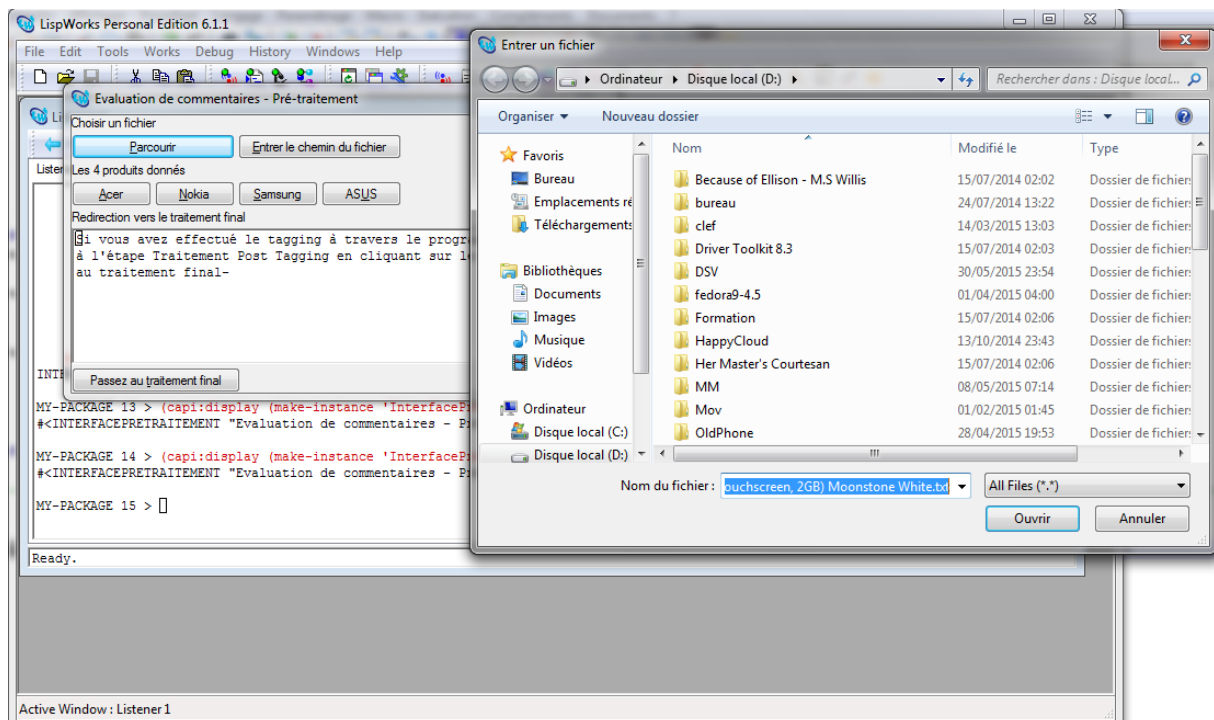


Figure 10: Interface de prétraitement

Le prétraitement s'effectue alors après validation. L'interface nous indique ensuite qu'il faut exécuter le code Java du Posttagging.

## 3. Interface de Post-Tagging

Après le prétraitement, on est redirigé vers l'interface de Post-Tagging, c'est-à-dire l'étape qui suit le prétraitement et le Posttagging. Dans cette étape, l'utilisateur a le droit de choisir entre deux versions : soit la version avec l'opinion satisfait et non satisfait seulement, soit avec l'opinion satisfait, non satisfait ou neutre. Le résultat est affiché avec un diagramme en bâtonnets comme suit :

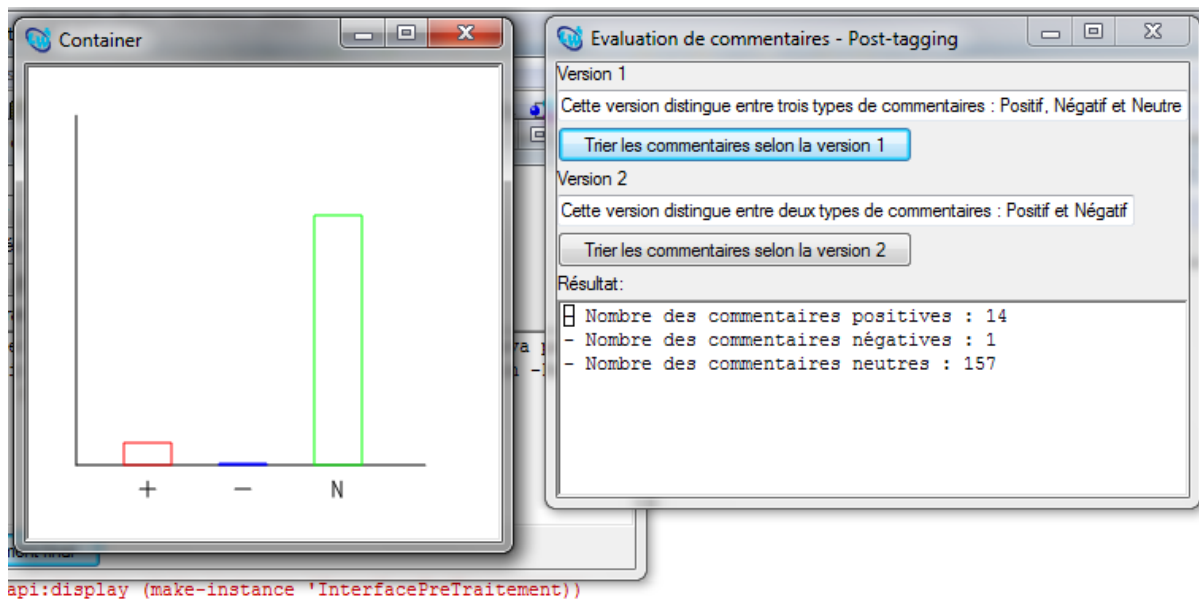


Figure 11: Interface de Post-Tagging: version 1 avec commentaire neutre.

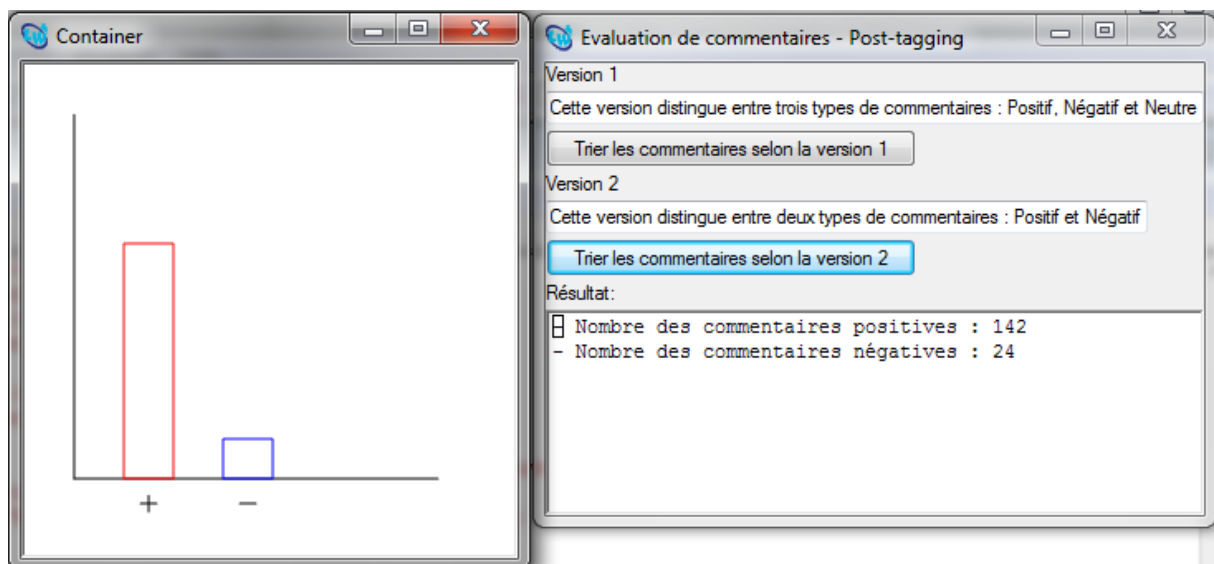


Figure 12: Interface de Post-Tagging: version sans commentaire neutre.

## Webographie

- [1] [https://fr.wikipedia.org/wiki/Opinion\\_mining](https://fr.wikipedia.org/wiki/Opinion_mining)
- [2] <http://stackoverflow.com/questions/15393797/lisp-splitting-input-into-separate-strings>
- [3] <http://cl-cookbook.sourceforge.net/win32.html>
- [4] <http://cl-cookbook.sourceforge.net/index.html> : utilisé pour trouver les fonctions adéquates lors du codage en Lisp.