HURST – GARABEDIAN
Master I Informatique 2015-2016

Analyse des sentiments au travers du texte

L’Analyse des Logiques Subjectives

L’A.L.S. est une méthode d’analyse des mots (lexèmes) pour en déduire la personnalité de l’auteur.

On prend en compte le sens des mots, non pas globalement, mais en le décomposant en "**atomes**" de sens le plus élémentaire possible afin de trouver des tendances générales ou **invariants subjectifs.**

Les Points de Vue

Parmi les nombreux points de vue exprimés à travers le vocabulaire d’une personne, notre étude s’est concentrée sur :

- **Le point de vue extraverti**
- **Le point de vue introverti**

Les Parlers

1. Le parler conservateur (I → I)
2. Le parler changement/destruction (E → E)
3. Le parler du progrès (E → I)
4. Le parler hésitant (I ou E)

Le Vocabulaire

Exemples :

Série A	Série B
Ouvert	Sage
Libre	Insensible
Bestial	Droit
Effrayant	Cérébral

- Les atomes de série A qui sont valorisés mettent en avant un point de vue extraverti. S'ils sont dévalorisés ils mettent en avant un point de vue introverti.

- Inversement les atomes de série B qui sont valorisés mettent en avant un point de vue introverti. S'ils sont dévalorisés ils mettent en avant un point de vue extraverti.

L’ordre des étapes

1. Suppression de la ponctuation et mise en minuscules
2. Lemmatisation des verbes
3. Suppression des locutions neutres
4. Suppression des expressions figées
5. Suppression des mots outils
6. Lemmatisation des noms et adjectifs
7. Elargissement des séries (Synonymie)
8. Analyse du point de vue

Ressources

- **Entrée :**
  - Listes d’atomes des séries A et B
  - Liste des mots outils avec leur type
  - Liste des expressions figées et locutions neutres
  - Liste d’expressions supplémentaires
  - Fichier de lemmatisation
- **Sortie :**
  - Fichiers contenant les mises à jour à faire aux listes de séries A et B

Structures de données - Tokeniseur

- **Arbre de Tokenisation (Tokeniseur)**

Liste à insérer dans l'arbre :

- arbre
- abris
- manger

Arbre qui en découle :

Annexe 2 : Exemple d'arbres de tokemisation

```
Liste à insérer dans l'arbre
    -avais
    -avait
    -fut
Arbre qui en découle :

A---------F
|         |
V         U
|         |
A         T
|         |
I        être
|
S-----T
|     |
avoir avoir
```

Structures de données – Tokeniseur et Listechar

Fonctions membres publiques

	Tokemiseur (String path) Crée un tokemiseur à partir d'un fichier. Plus de détails...
void	**tokemFile** () Lancement de la tokemisation.
void	**createTree** (String line) Permet d'ajouter un mot dans l'arbre. Plus de détails...
void	**createVthree** (String line, String inf) Permet d'ajouter un verbe dans l'arbre. Plus de détails...
String	**findTokem** (String s) Cherche si le tokem est dans l'arbre lexical. Plus de détails...

Fonctions membres publiques

void	**setIsInit** (boolean init)
void	**setChar** (char c)
void	**setFils** (**Listechar** fils)
void	**setFrere** (**Listechar** frere)
boolean	**getIsInit** ()
char	**getChar** ()
Listechar	**getFils** ()
Listechar	**getFrere** ()
String	**getFinal** ()

Structures de données – Liste de lemmatisation

- **Liste de lemmatisation**
  - Une liste de Strings non-lemmatisés
  - Un String représentant la forme lemmatisée

Fonctions membres publiques

	ListeLemm () Initialise une liste de lemmatisation vide.
	ListeLemm (String lemm) Crée une liste de lemmatisation vide avec une forme lemmatisée. Plus de détails...
	ListeLemm (ArrayList< String > A, String lemm) Crée une liste de lemmatisation à partir d'une liste de String et d'une forme lemmatisée. Plus de détails...
void	**add** (String mot) Ajoute un mot à la liste de lemmatisation. Plus de détails...
void	**add** (String mot, String l) Ajoute un mot à la liste de lemmatisation et définit la forme lemmatisée. Plus de détails...
void	**setLemm** (String lemm) Modifie la forme lemmatisée. Plus de détails...
String	**getLemm** () Renvoie la forme lemmatisée. Plus de détails...
void	**setArray** (ArrayList< String > A) Modifie la liste de mots non-lemmatisés. Plus de détails...
ArrayList< String >	**getArray** () Renvoie la liste de mots non-lemmatisés. Plus de détails...

Structures de données – Liste de lemmatisation

Une structure supplémentaire a été créée pour faire une liste de listes de lemmatisation (ListeLemmWrapper)

Fonctions membres publiques

void	**add** (String mot, String l) Ajoute un mot dans la liste de lemmatisation au bon emplacement si une liste existe, sinon elle crée une nouvelle liste. Plus de détails...
ArrayList< **ListeLemm** >	**getArray** () Renvoie les listes de lemmatisation. Plus de détails...
ListeLemm	**find** (String l) Renvoie la liste de lemmatisation pour une forme lemmatisée donnée. Plus de détails...
void	**display** () Affiche tous les éléments des listes de lemmatisation.

Structures de données

- **Arbres de Tokenisation**
  - Avantages : Rapidité lors de la recherche et utilisation de peu de mémoire (O(n*m))
  - Inconvénients : Un peu plus lent qu'une liste à créer et impossibilité d'itérer sur les mots
- **Listes de lemmatisation**
  - Avantages : Possibilité d'itérer sur les éléments de la liste, et la création de la liste ainsi que l'ajout d'éléments sont rapides
  - Inconvénients : Beaucoup plus lent lors de la recherche que le tokeniseur sur une grande liste de mots

Lemmatisation

Fonctions membres publiques

	Lefff (String path) Constructeur pour le lefff. Plus de détails...
void	**readLefff** () Permet la lecture d'un fichier de lefff Permet de lire un fichier de lefff formaté sous la forme : "abaissé v abaisser" afin de préparer la mise à l'infinitif des verbes.
void	**readOutil** (String path) lit et stock dans un arbre de tokemisation la liste des mots outils Plus de détails...
void	**readExp** (String path) permet de charger une liste d'expression Plus de détails...
void	**writeFile** (String txt, String titre) Ecrit le nouveau texte dans un fichier. Plus de détails...
String	**traiteVerbe** (String oldTexte) permet de traiter le texte traite le texte ce trouvant à l'adresse envoyé en paramètre Plus de détails...
String	**traiteNomsP** (String oldTexte) Lemmatise tous les noms propres. Plus de détails...
String	**traiteNoms** (String oldTexte) Lemmatise tous les noms communs. Plus de détails...
String	**traiteAdj** (String oldTexte) Lemmatise tous les adjectifs. Plus de détails...
String	**traiteNetAdj** (String oldTexte) Lemmatise tous les noms et adjectifs. Plus de détails...
String	**findExp** (String line) vérifie si il y a des expressions à ne pas traiter pour la lemmatisation des verbes Plus de détails...
String	**traiteTexte** (String p) permet de traiter un texte Plus de détails...
String	**supprExpNeutre** (String txt) enlève toute les expressions neutres du texte Plus de détails...
String	**supprPonctuation** (String txt) Supprime la ponctuation dans le texte. Plus de détails...
String	**supprOutils** (String txt) Supprime les mots outils. Plus de détails...
String	**openTexte** (String path) ouvre un texte, et le stock dans un String Plus de détails...
Tokemiseur	**getAdj** () Renvoie l'arbre des adjectifs. Plus de détails...

Fonctionnalités :

- Suppression de mots et de ponctuation
- Lemmatisation des verbes, noms et adjectifs
- Opérations sur des fichiers (écriture et lecture)

Synonymes

Fonctions membres publiques

	Synonymes (String lang) Construit un objet **Synonymes** avec une langue spécifique. Plus de détails...
	Synonymes () Construit un objet **Synonymes** avec la langue par défaut (fr_FR)
boolean	**findSynonymes** (String mot) Retrouve les synonymes d'un mot grâce à l'API de thesaurus.altervista.org et les stocke dans la liste de synonymes S. Plus de détails...
void	**classification** (**Tokemiseur** SerieA, **Tokemiseur** SerieB) Détermine selon les synonymes trouvés si le mot d'origine appartient à la série A ou série B puis l'écrit dans un fichier de mise à jour serieA_maj.txt ou serieB_maj.txt. Plus de détails...
boolean	**dejaClassifie** (String mot) Renvoie si le mot a déjà été classifié grâce aux synonymes ou non. Plus de détails...
void	**classifierMot** (String mot, **Tokemiseur** SerieA, **Tokemiseur** SerieB) Détermine si un mot appartient à la série A ou à la série B. Plus de détails...
void	**classifierListe** (List< String > L, **Tokemiseur** A, **Tokemiseur** B) Permet de déterminer à quelles séries appartiennent tous les mots d'une liste. Plus de détails...
void	**classifierTableau** (String[] T, **Tokemiseur** A, **Tokemiseur** B) Permet de déterminer à quelles séries appartiennent tous les mots d'une liste. Plus de détails...
void	**findSynonymes** (String mot, String lang) La même chose que **findSynonymes(String mot)** mais avec une option de langue. Plus de détails...
ArrayList< String >	**getSynonymes** () Renvoie la liste de **Synonymes**. Plus de détails...
String	**getMot** () Retourne le mot d'origine pour lequel les synonymes ont été trouvés. Plus de détails...
void	**setLang** (String lang) Permet de régler la langue. Plus de détails...
String	**getLang** () Renvoie la langue actuelle. Plus de détails...
void	**setAdj** (**Tokemiseur** arbreAdj) Permet de règler l'arbre d'adjectifs. Plus de détails...
Tokemiseur	**getAdj** () Renvoie l'abre des adjectifs. Plus de détails...

Fonctionnalités :

- Récupération de synonymes via une API
- Classification des molécules grâce aux synonymes

Conclusion

- **Ce qui est fait :**
  - Nous avons réussi à trouver les structures de données optimales
  - Une grande partie de la lemmatisation est faite
  - La classification des molécules grâce aux synonymes est efficace malgré une API moins complète
- **Ce qui reste à faire :**
  - Règler certains problèmes issus de la nature du fichier de lemmatisation (doublons)
  - La valorisation des mots du texte
  - L’analyse du point de vue