# Rapport de projet

## Introduction

Afin de finir l’année de Master 1 Informatique, il nous a été demandé de faire un projet parmi une liste de sujet, ou un Stage. Julian Hurst et moi-même avons eu la chance de tomber sur l’un des sujets que nous avions choisis.

Nous nous sommes donc vue attribué le Sujet *« Analyse des sentiments à travers le discours ».* Malheureusement, dut à des soucis d’emplois du temps qui ne concordait pas, le projet n’a commencé que tardivement, une fois les examens finis. Une fois le premier rendez-vous effectué, nous avons pu démarrer le projet.

Afin de réaliser ce projet, nous étions Madame Line Jakubiec, Monsieur Vincent Risch et Monsieur Jean-Jacque Pinto qui est l’initiateur de la méthode de l’ALS (Analyse des Logiques Subjectives)

La première étape été de nous mettre d’accord sur les standards utilisé lors du projet. Pour le nommage nous avons choisi le standard $$$. Pour ce qui est des commentaires, en plus des commentaires à proximité de certaines actions, nous avons choisi de prévoir des commentaires compatibles avec Doxygen afin de pouvoir générer une doc.

Pour ce qui est de l’arborescence, nous avons choisi d’appeler avec un fichier Main.java les fichier java contenue dans le package affichage. Pour le moment, seul un fichier Als.java s’y trouve. Les classes appartenant au package affichage s’occupe de l’affichage du menue contextuel, et des résultats du traitement.

Chaque module possède quant à eux un package qui leur est propres, pour l’instant ils sont au nombre de 3. Le lemmatiseur qui permet de faire la lemmatisation du texte, le Tokemiseur qui s’occupe de créer un arbre de tokem afin d’optimiser la recherche de mots pour certaines étapes, et synonyme qui permet de trouver les synonymes d’un mot qui n’est pas encore classifié. Afin de disposer du schéma complet, reportez-vous à l’annexe 1.

Le projet quant à lui a avancé plutôt de façons itératives celons le feedback fait lors des réunions à Luminy. Ces réunions été l’occasion de soulever des problèmes de traitement dû à l’informatique, et de réfléchir à des solutions pour les contourner.

## Description du sujet

Notre sujet été l’Analyse des Sentiments au travers du discours. Ce sujet ce base sur la méthode de l’ALS (Analyse des Logiques Subjectives) initié par Monsieur Jean-Jacques Pinto. Cela consiste à dégager des sentiments au travers du discours d’une personne au fil d’un processus en plusieurs étapes.

Dans notre cas nous avons travaillé sur des textes, mais il est possible d’étendre ce procédé au discours oral car celui-ci celons l’ALS répond aux mêmes règles.

Lors de l’explication du sujet, nous prendrons pour exemple le texte de Gombrowicz suivant : « Ma morale, c’est d’abord de protester au nom de mon humanité personnelle, d’exprimer ma révolte à coup d’ironie été de sarcasmes. Je suis un humoriste, un plaisantin, je suis un acrobate et un provocateur, je suis un cirque, poésie, horreur, bagarre, jeu, que voulez-vous de plus ? »

L’ALS permet donc d’analysé un texte comme le texte précédent, et d’en ressortir la personnalité de son auteur. Pour ce faire, l’ALS fait une analyse sur les mots (Lexèmes) non pas sur son sens globale, mais en le décomposant en « atomes » de sens le plus élémentaire possible.

Un atome, est un adjectif qui exprime des propriétés simples (ex : ouvert/fermé) que l’on peut classer celons deux séries opposées.

* La série A concerné l’extérieur, le changement, le désordre, la destruction de l’ancien. Elle se compose d’adjectifs simples comme : ouvert, souple, varié, changeant, nouveau, libre…
* La série B concerne au contraire l’intérieur, le non-changement, l’ordre, la conservation. Elle se compose d’adjectifs simples comme : sérieux, ferme, stable, ancien, solide, durable…

Les mots complexes, ou molécule quant à eux sont des adjectif complexes, des noms, des verbes, et des adverbes dont le sens peut se décomposer en atome A ou B.

Afin de les classer, on se sert de la proximie d’un mot. Pour ce faire, on récupère tous les mots proches du mot que l’on veut classifier, et on calcul la somme de ses atomes avec -1 pour la série B, et -1 pour la série A. Si le résultat de cette somme est de 20% ou plus du totale des mots de la proxémie, alors on peut le classer. Dans le cas contraire, on classe le mot en tant que neutre.

Cette classification ne permet pas encore de donner un point de vue car les mots ne sont pas valorisés.

### Le point de vue

Le point de vue dans l’ALS est une classification valorisée du discours. Si les mots de la série A sont valorisés (+) et ceux de la série B dévalorisés (-), alors le point de vue sera extraverti, et inversement, si les mots de la série A sont dévalorisés (-) et ceux de la série B valorisés (+) alors le discours sera introverti.

Par conséquence, une personne extravertie utilisera les mots de la série A pour parler de ce qu’il aime, et de la série B pour parler de ce qu’il critique. Au contraire, une personne introvertie choisira des mots de la série B pour parler de ce qu’il aime, et des mots de la série A pour critiquer ce qu’il n’aime pas.

### Les parlers

Les parlers est l’extension à l’échelle d’une vie du point de vue, et permet d’en déduire quatre parlers différents.

* Le premier est le parler **« conservateur »** **(I → I)**, correspondant à la personnalité obsessionnelle « introvertie incorruptible » nostalgique du paradis perdu, qui commence introvertie, et finit introvertie.
* Le deuxième est le parler **« changement/destruction » (E** **→ E)**, correspondant à la personnalité hystérique : « extraverti incorrigible, tenté par l’enfer, qui commence Extravertie, et finit Extravertie.
* Le troisième est le parler **« du progrès » ou « constructeur » (E → I)**, sans équivalent séméiologique : « extravertie repentie » transitant par le purgatoire, qui commence Extravertie, et finit Introvertie.
* Le quatrième est le parler **« hésitant » (I ou E)**, en gros la personnalité phobique : « l’éternel indécis », oscillant toute sa vie entre Extravertie, et Introvertie.

### Hypothèse de l’ALS

C'est le discours parental qui détermine, non de façon linéaire mais avec des transformations elles-mêmes « programmées », le discours fantasmatique de l'enfant, différemment selon qu'il est idéalisé ou rejeté (cas extrêmes). L'enfant, une fois identifié au texte du désir parental, qualifiera et traitera désormais tout objet (y compris lui-même et son parent) comme le parent l'a qualifié et a souhaité le traiter. C'est la satisfaction du parent, et non la sienne, qu'il exprime et recherche sans le savoir. Les adjectifs extraits des appréciations du parent sur lui, et les verbes décrivant le sort qu’il lui souhaite, fourniront les atomes valorisés dans les énoncés fantasmatiques, et constitutifs des séries.

### Traiter un texte

Reprenons notre texte texte de Gombrowicz: « Ma morale, c’est d’abord de protester au nom de mon humanité personnelle, d’exprimer ma révolte à coup d’ironie et de sarcasmes. Je suis un humoriste, un plaisantin, je suis un acrobate et un provocateur, je suis un cirque, poésie, horreur, bagarre, jeu, que voulez-vous de plus ? »

Pour appliquer l’ALS il faudra respecter un certains nombres d’étapes intermédiaires afin de pouvoir analyser les lexèmes.

#### Lemmatisation des verbes

Dans un premier temps, il faudra lemmatiser le texte afin d’en faire ressortir les lexèmes. Tout d’abord, il faudra remplacer les verbes par leur infinitif. Cette étape traitera les verbes est, été suis, suis, suis, voulez. Cela donnera le texte suivant :

« Ma morale, c'être d’abord de protester au nom de mon humanité personnelle, d'exprimer ma révolte à coups d'ironie et de sarcasmes. Je être un humoriste, un plaisantin, je être un acrobate et un provocateur, je être cirque, poésie, horreur, bagarre, jeu, que se vouloir-vous de plus ? »

#### Suppression des locutions neutres

Ensuite, il faut supprimer les locutions neutres à l’aide d’une liste. Dans notre texte, il faudra supprimer d’abord, au nom de, à coup d’. On obtiendra le texte suivant :

« Ma morale, c'être de protester mon humanité personnelle, d'exprimer ma révolte ironie et de sarcasmes. Je être un humoriste, un plaisantin, je être un acrobate et un provocateur, je être cirque, poésie, horreur, bagarre, jeu, que se vouloir-vous de plus ? »

#### Recherche des expressions figées

Ensuite, il faut chercher les expressions figées afin de les traiter dans leur ensemble, et de les classer dans la série A ou B. Dans ce texte, il n’y en a pas, par conséquence, il reste le même que précédent.

#### Lemmatisation des noms et adjectifs

Dans cette étape, il faudra mettre sous leur forme lemmatisée les nom et adjectif afin de pouvoir les analyser. Cela donnera le texte suivant :

« Ma moral, c’être de protester mon humanité personnel, d’exprimer ma révolte, ironie et de sarcasme. Je être cirque, poésie, horreur, bagarre, jeu, que vouloir-vous de plus ? »

#### Suppression des mots outils

Maintenant, il faut supprimer tous les mots outils et la ponctuation qui n’apportent rien à l’analyse afin qu’ils ne soient pas traités. On obtient au final le texte suivant :

« Moral protester humanité personnel exprimer révolte ironie sarcasme cirque poésie horreur bagarre jeu vouloir plus »

#### Analyse mot à mot

Pour l’analyse de chaque mot, si le mot est dans une série, le classé dans la série correspondante

Si le mot n’est pas dans la liste tel que le mot morale, il faut procéder par proxémie. Pour cela on prend tous les mots de sa proxémie celons le site du CNRTL, et qui trouve une correspondance dans le tableau des atomes classés.

« beau, bon, chaste, convenable, correct, digne, droit, décent, exact, honnête, honorable, intègre, juste, parfait, probe, propre, pur, respectable, régulier, sage, sérieux, vertueux ». Tous ces mots étant de série B, on a un total de +22, ce qui représente 100%, moral est donc de série B.

Ensuite, il faudra valoriser le résultat. Par exemple, pour morale on se retrouve avec un mot de série B valorisé, donc B+, il impliquera donc l’introversion.

#### Conclusion

Une fois tous les mots traités un à un, il sera possible de dégager un point de vue général au texte.

### Le développement

Nous sommes les premiers à travailler sur ce projet, par conséquent, il n’y avait pas d’existant en termes de développement. Pour ce qui est de l’existant autre, nous avions accès à un résumé de ce qu’est l’ALS rédigé par Monsieur Pinto.

Pour le traitement de texte, nous avions aussi à notre disposition des listes de mots pour les mots outils, les locutions neutres, les expressions figé. Pour finir, afin de lemmatiser les textes nous avions aussi accès à un fichier de lemmatisation mis à disposition par Madame Line Jakubiec.

A l’heure où j’écris ces lignes, le programme comporte la lemmatisation du texte, et sont traitement afin de pouvoir en faire ressortir les mots intéressants à analyser pour l’ALS.

#### Général

Comme expliqué plus haut, nous avons choisi une arborescence simple sous forme de modules qui sont tous géré par le Package affichage (voir annexe 1).

Une fois le programme lancé, tout est géré dans « package/Als.java ». Pour ce faire, pour le moment, il y a un menue contextuel console fait via une boucle. Pour quitter le programme, il suffit de taper « quit » lors du premier message, à savoir de l’action à effectuer.

Pour le moment, le menue permet de traiter un texte pré-chargé, ou de rentrer le chemin d’accès d’un texte au format txt.

Ce choix par module a été fait car il simplifie les ajouts, et permet au programme d’être évolutif, et facilement améliorable, et modifiable.

#### Affichage

Le package affichage s’occupe de l’affichage qui pour le moment n’est que via console, ainsi que de la gestion du processus celons les choix de l’utilisateur.

Lors de son lancement, il s’occupe de charger tous les fichiers texte utile au traitement afin de ne pas avoir à les recharger à chaque texte. Tant que le programme est lancé, les fichiers textes pour le traitement ne seront donc pas rechargé.

Une fois tous les fichiers nécessaires chargé, il affiche le menue textuel à l’utilisateur. Le premier choix est l’action à faire, soit traiter un texte, soit quitter l’application. Dans le cas où l’utilisateur choisit de poursuivre, il pourra choisir d’entrer le chemin d’accès de son propre texte, ou de choisir un texte dans un choix de 4 textes. Afin de faciliter l’ajout de texte dans cette liste, pour le choix, le programme fait appel à une méthode choixTexte. Les textes doivent être placé dans ./res/txt/.

#### Tokemisation

Le package tokemisation s’occupe de stocker les différentes listes, mais aussi de rechercher de façons optimisées dans ces listes. Il utilise les connaissances acquises lors la matière FIL avec Monsieur Ramisch.

Pour ce faire, le programme créée un arbre de tokem dans lequel il ajoute un mot à chaque utilisation de la méthode createVthree(String line, String inf). Bien qu’elle soit à l’origine créée pour traiter les verbes, elle peut être utilisé dans de nombreux cas.

Pour ce qui est de l’arbre, chaque feuille représente une lettre. Lors de l’ajout de la première branche, il suffit de mettre les lettres du mot courant une à une en tant que fils de la lettre qui la précède, et de terminer par une valeur de fin. Dans le cas des verbes, on termine en stockant le verbe à l’infinitif.

Lorsque ce n’est pas le premier mot, tant qu’on a une concordance de lettre, on descend dans l’arbre. Lorsque la lettre diffère, on cherche dans les frères si la lettre courante est stockée, si c’est le cas, on continue en cherchant les fils, puis frère jusqu’à trouver le mot complet, sinon on crée un frère, et on crée ses fils.

Dans le cas où le mot courant serait un mot similaire à un précédent mot, mais plus court (exemple : morale et moral), une fois à la fin du mot, on vérifie de bien avoir la valeur de fin. Si on n’a pas la valeur de fin, on parcourt les frères jusqu’à en trouver un qui n’a pas de frère, et on crée un frère contenant la valeur de fin. (Voir annexe 2 pour l’arbre)

Afin de créer cet arbre, nous avons mis en place une classe qui joue le rôle de structure de feuille. La classe Listchar permet donc de stocker une lettre, un fils, un frère, et une valeur final de type String. A part cela, elle contient juste des accesseurs, et des mutateurs, ainsi que 2 constructeur avec paramètre. Un pour créer une feuille caractère, et un pour créer une feuille finale.

Pour ce qui est de la recherche, le principe est assez similaire à celui de l’ajout d’un mot. On descend donc dans l’arbre tant que le caractère courant est identique dans l’arbre. Si le caractère est différent, on cherche un frère identique, et dans le cas où l’on trouve un frère, on recommence à descendre de fils en fils comme précédemment.

Si on arrive à la fin du mot courant avant d’atteindre une valeur finale de l’arbre, on vérifie dans les frères s’il n’y a pas une valeur finale.

Dans le cas où on ne trouve plus de frère qui concorde avec la lettre suivante, alors, on renvoie null en signe de négation à la recherche, sinon, on renvoie la valeur finale trouvé.

Ce type d’arbre permet une recherche rapide et efficace, et ceux même dans les grandes listes de mot. Le seul inconvénient c’est que pour l’heure, sa création est un peu plus couteuse que celle d’une liste, mais lors de chargement de grand texte, ou de plusieurs textes à la suite, le temps gagner dans la recherche de mots est largement supérieur à la perte lors de la création.