**

Projet de ZZ2

*Nuage de mots*

Extraction statistiques sur le contenu de documents.

BARBESANGE Benjamin – GARÇON Benoît

Tuteur : D. HILL — Référent : V. BARRA

01/03/2016

Table des matières

[Introduction 2](#_Toc437262082)

[Nuage de mots 3](#_Toc437262083)

[Présentation 3](#_Toc437262084)

[Applications 3](#_Toc437262085)

[Recherches de l'existant 3](#_Toc437262086)

[Solution présentée 4](#_Toc437262087)

[Les données 4](#_Toc437262088)

[Le programme 4](#_Toc437262089)

[Description 4](#_Toc437262090)

[Fonctionnement 4](#_Toc437262091)

[Le code 4](#_Toc437262092)

[Principe 4](#_Toc437262093)

[La table de hachage 4](#_Toc437262094)

[Outil auxiliaire : génération de phrases 5](#_Toc437262095)

[Intérêt 5](#_Toc437262096)

[Le programme 5](#_Toc437262097)

[Présentation 5](#_Toc437262098)

[Utilisation 5](#_Toc437262099)

[Elaboration du programme 5](#_Toc437262100)

[Principe 5](#_Toc437262101)

[Code 5](#_Toc437262102)

[Conclusion 6](#_Toc437262103)

# Remerciements

Dans le cadre de ce projet, nous remercions particulièrement D. Hill notre tuteur de projet, ainsi que V. Barra référent de ce projet.

# Résumé

Version française.

Mots-clés : nuage, mot.

# Abstract

English version.

Key words: tag, cloud.

# Introduction

Ce projet nous a été proposé par l’Institut Supérieur d’Informatique de Modélisation et de leurs Applications (ISIMA), dans le cadre de la seconde année du cycle d’ingénieur.

Le thème de ce projet est l’analyse et l'extraction statistique de texte en français. Par cet intitulé, nous avons pris la direction des nuages de mots, que l'on peut facilement associer aux mots clés d'un texte. Ils sont généralement rencontrés sous forme graphique et contiennent les mots les plus répandus d’un document texte (livre, page Web, article, …). Ces mots apparaissent dans une police plus ou moins grande selon leur fréquence d’apparition. C’est ainsi que d’un rapide coup d’œil nous pouvons dégager un thème ou bien encore une tendance générale dans un document.

Cet outil peut avoir de nombreuses utilisations dans le domaine du référencement de documents par exemple ou encore dans la vérification d'écrits scientifiques. On peut également trouver une utilité esthétique lorsqu’il est possible d’avoir un rendu graphique.

Certaines solutions existent déjà sur le Web, mais il apparaît que certaines ne répondent pas à certaines exigences comme par exemple la lecture de fichiers autre que le format de texte brut txt ou encore la possibilité d’importer sa propre liste de mots à ignorer. En effet la plupart des produits existant étant anglophone, les nuages de mots générés à partir de textes francophones sont vite pollués de ‘le’, ‘la’, etc.

C’est pourquoi dans ce projet, nous allons nous efforcer dans un premier temps de tester différentes structures permettant d’extraire les mots les plus utilisés d’un fichier texte tout en spécifiant un fichier contenant une liste de mots à ignorer. Cette phase doit pouvoir se faire en un temps acceptable (1min maximum) sur un fichier volumineux (200 000 lignes ou 500 pages). Une fois cette étape terminée nous implémenterons le support des fichiers pdf qui est le format le plus répandu de documents mis en forme, puis nous ferons de même pour le support de pages Web. Enfin nous nous initierons à la programmation d’interface graphique en Qt ainsi qu’en Web pour avoir un rendu graphique de nos résultats.

# Nuage de mots

## Présentation

Un nuage de mots est une représentation graphique simplifiée d’un texte cohérent. Il regroupe l’ensemble des mots les plus représentatifs du texte en question et organise une hiérarchie de ceux-ci grâce à un jeu de polices et de couleurs. L’extraction des mots-clés peut se faire simplement par analyse statistique du nombre d’occurrences de chaque mot dans le texte ou bien par analyse sémantique.

Historiquement, les nuages de mots sont utilisés à des fins publicitaires et commerciales. Des outils en lignes permettent de générer de véritables visuels professionnels transformant le nuage de mots en un produit commercial. Ils sont aussi utilisés sur le Web pour présenter les recherches les plus effectuées sur un moteur pour que l’utilisateur puisse atteindre plus rapidement les résultats les plus populaires.

## Applications

Ce projet peut avoir de nombreuses applications. Certaines peuvent être purement esthétiques pour la présentation simple d'un site Web ou d'un document. En effet, le nuage de mot va sortir les mots les plus utilisés, ce qui permet d'avoir la tendance de la ressource concernée de manière plus ou moins esthétique en fonction de ce qui est utilisé (ce que nous présentons dans la partie suivante).

Une autre application plus pratique concerne l'indexation des documents. Il est tout à fait possible d'imaginer que lorsque l'on place une ressource en ligne (un rapport de projet, un article scientifique) on souhaite en dégager des mots clés qui serviront ensuite à retrouver plus facilement cette ressource.

Enfin une dernière application plus spécifique concernerait les écrits scientifiques. Lorsqu'un article est rédigé par un scientifique sur un domaine précis, il y a certains termes que l'on attend comme dominants. Il dont possible d'extraire les mots les plus utilisés dans cet article et de vérifier si les termes attendus ressortent ou non. S'ils ne ressortent pas, il est possible d'imaginer que l'article ne concernera pas le sujet et ainsi ne le présentera pas correctement. Ceci peut permettre de gagner du temps lors de la vérification des écrits scientifiques.

## Étude de l'existant

De nombreux outils existent déjà et permettent de créer des nuages de mots. Après investigations, il est apparu les solutions suivantes.

La plupart des sites se servent d'une zone de texte dans laquelle on colle le texte d'où on veut extraire le contenu. On peut également fournir un lien vers le flux RSS d'un site, ce qui va généralement en donner la tendance.

Le site le plus connu est http://www.wordle.net/. Il permet de créer un nuage de mots à partir d'un texte que l'on colle sur le site. Puis on obtient un résultat graphique, il est important de noter que ce site ne sera bientôt plus compatible puisqu'il repose sur les applet Java qui ne seront bientôt plus supportés par les navigateurs. Un autre site permettant de faire ceci est http://www.tagxedo.com/. Le point faible de ces sites et qu'ils reposent sur l'utilisation de plugins et ainsi ne sont pas compatible avec tous les navigateurs.

Certains de ces sites proposent des formes prédéfinies de nuage, et d'autres encore permettent de créer ses propres formes.

Le site http://tagcrowd.com/ possède un avantage par rapport aux autres sites dans la mesure où il est possible d'importer un fichier (au format texte) et d'y extraire le nuage. Voici un exemple crée à partir d'un compte rendu de TP Système :



Figure 1 - Nuage de mot à partir de www.tagcrowd.com

Comme on peut le voir, nous retrouvons facilement de thème du compte rendu qui concernait la gestion des droits sur une machine.

Notons que beaucoup de ces outils sont conçus pour tirer parti des informations venant de textes en Anglais. C'est pourquoi les pluriels de certains mots français ne sont pas gérés.

# Solution présentée

Les données occupent toujours une place prépondérante dans un programme. C’est pourquoi nous allons dans un premier temps parler des données que nous manipulons dans le programme. Ensuite nous détaillerons comment se servir du programme pour enfin en décrire les mécanismes.

## Les données

La présentation d'un nuage de mot ne représentait qu'un bref aperçu du mécanisme complexe qui mène à sa réalisation. En effet, le simple fait d'observer les mots les plus courants d'un texte ne suffit pas uniquement à en dégager une tendance. Il faut également prendre en compte d'autres paramètres en plus de la donnée principale.

### Donnée principale

La principale donnée de notre programme va être le texte à analyser. Ici, nous allons gérer les formats de fichiers .txt et .pdf, car ce sont des fichiers de formats simples et largement utilisés, autant par la communauté scientifique que par le grand public.

### Données annexes

Un premier paramètre à prendre en compte dans notre étude concerne la langue du texte. En effet, en fonction de la langue utilisée, il est nécessaire d’utiliser un codage de caractères approprié afin que ceux-ci ressortent de la bonne manière à l’écran. Ici, nous utilisons un codage UTF-8, ce qui assure l’utilisation de langues comme l’anglais, le français, l'allemand ou encore l’italien. D’autres langues comme le russe ou le chinois par exemple utilisent des caractères spéciaux qui ne seront pas pris en charge.

La seconde chose à prendre en compte est que dans une langue, beaucoup de mots sont utilisés pour assurer la cohérence d’un texte, comme par exemple des pronoms personnels, démonstratifs ou encore des conjonctions de coordinations. Si nous n'ignorons pas ces mots lors de la lecture, nous pouvons être surs qu'ils seront largement dominants dans le retour du programme. C’est pourquoi nous devons utiliser une liste de mots à ignorer dans le texte. Cette liste sera fournie au format texte avec l’extension “.config” et à raison d’un mot par ligne dans ce fichier.

Voici un exemple d’affichage des 10 mots les plus répandus du livre de Simulation de D. Hill, avec et sans liste de mots à ignorer :

|  |  |
| --- | --- |
| Sans ignore.PNG  Figure - Résultat sans liste de mots | Avec ignore.PNG  Figure - Résultat avec liste de mots |

Comme on peut le remarquer, lorsqu’aucune liste de mots à ignorer n’est spécifiée, nous obtenons beaucoup de mots qui n’aident pas à avoir un aperçu du texte. Lorsque l’on ajoute la liste de mots à ignorer, nous retrouvons beaucoup plus de termes explicites. Il va cependant toujours rester certains mots qui n’apportent pas de précisions sur le texte. Ainsi plus la liste des mots à ignorer sera complète, plus le résultat final sera focalisé sur le sens du texte.

Enfin la dernière chose à laquelle il faut penser est qu’un texte s’organise autour de ponctuation. Nous disposons donc d’un autre fichier de configuration au format texte ayant l’extension “.config” et qui va contenir les caractères de ponctuation les uns après les autres sur la première ligne de ce fichier. Ces caractères nous permettrons par la suite de découper notre document en mots.

## Le programme

### Description

Le programme principal peut s'utiliser de différentes manières :

* En mode texte, ce qui permet des interactions directes dans la console et va fournir le résultat à l'écran ou dans un fichier si on redirige la sortie standard du programme
* En mode Web, via une page de navigateur
* En mode graphique, qui l'utilisation aisée du programme et produira des affichages directement dans le fenêtre

Quel que soit le mode choisi, il faudra toujours renseigner certaines données :

* Le fichier à traiter
* Le fichier contenant les séparateurs, au format décrit dans la section précédente
* Un fichier contenant des mots à ignorer dans le texte, au format décrit dans la section précédente

Un fichier de séparateurs et de mots à ignorés est fourni de base dans le programme et concerne la langue Française. Il n’est donc pas nécessaire de les spécifier.

Une fois ces renseignements donnés, le programme va procéder à l'analyse du texte et fournir les résultats selon le mode dont on l'utilise.

### Mode texte

C’est le mode le plus simple du programme. Il suffit de lancer le programme dans un terminal en utilisant la commande : ./NuageDeMots. Cette commande va lancer le programme avec le test par défaut, et produire directement l’affichage. Il n’y a ici aucune interaction de la part de l’utilisateur.

Pour interagir avec le programme, il faut le lancer avec l’option -v : ./NuageDeMots -v. Cette option va permettre d’obtenir le menu suivant :

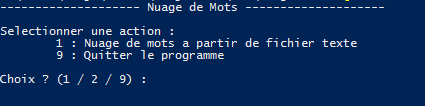


Figure 2 - Menu de l’interface texte

Nous sommes ensuite invités à faire un choix en entrant ‘1’ pour lancer le programme, ou ‘9’ pour le quitter.

Lorsque le choix ‘1’ est sélectionné, il est demandé d’entrer successivement le chemin vers le fichier de mots à ignorer, le chemin vers le fichier de séparateurs (si ces deux paramètres sont laissés vides, on utilise les fichiers par défaut), puis enfin le fichier que nous souhaitons traiter (si le fichier n’est pas renseigné, nous utilisons le fichier exemple fourni avec le programme).

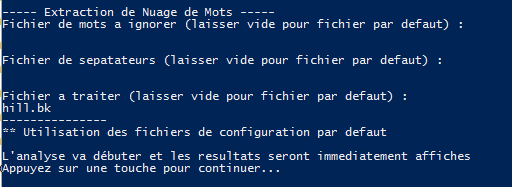


Figure 3 - Menu de sélection des fichiers

Si le chemin vers un fichier n’est pas le bon, nous utilisons le fichier par défaut correspondant. Un affichage indique également quels sont les fichiers par défaut utilisés.

Une fois tous les paramètres renseignés, le programme indique qu’il est prêt à s’exécuter. Il faut alors appuyer sur une touche et attendre la fin du traitement. L’affichage se fera directement dans le terminal.

Lorsque le traitement est terminé, le programme affiche à nouveau le menu principal. Il est alors possible de traiter un autre fichier ou de quitter.

### Mode Web

En cours...

### Mode interface utilisateur

En cours...

## Le code

Pour élaborer cette solution, nous avons dû mettre en place un certain nombre de structures que nous décrirons ici. De plus, d'autre mécanismes ont été introduits et seront également présentés dans cette partie.

### Principe

Le principe général du programme est le suivant :

1. Chargement des différents fichiers nécessaires
2. Lecture des mots du fichier principal
   1. Vérification de la présence dans la liste à ignorer
   2. S’il ne faut pas l’ignorer on le compte et le stocke dans une structure
3. Tri de la structure contenant les mots en fonction du nombre d’apparition
4. Affichage du résultat du tri

### La table de hachage

C’est la première structure que nous avons pensé à implémenter car il est facile d’obtenir un code de hachage via une chaîne de caractères. Nous allons stocker des éléments constitués :

* Du mot inséré, dans une chaîne de caractère
* Du nombre d'occurrences de ce mot, sous la forme d’un entier non signé

Cette structure permet d’avoir des insertions rapides dans une table. Cependant celle-ci doit avoir une taille suffisante pour ne pas devoir gérer trop de collisions ce qui pourrait entraîner une baisse des performances de la structure.

La fonction de hachage que nous avons utilisée repose sur l’algorithme “dj2b” :

1. On initialise le hash à 5381
2. Pour chaque caractère c du mot
   1. On multiplie le hash par 33
   2. On ajoute le code du caractère c au hash
3. On effectue le modulo du hash par la taille de la table (pour ne pas dépasser)
4. On retourne le hash

Les valeurs dans cette fonction ont été choisies de manière à ce qu’il y ait le moins de collisions possible dans une table (à condition que la table ait une taille suffisante).

Grâce à cette structure, nous pouvons localiser l’emplacement d’un mot en O(1). Il convient ensuite de le chercher dans une liste gérant les collisions, recherche qui se fait en O(n), n étant le nombre de mots de cette liste. Le choix de l’algorithme de hachage est donc très important et va conditionner le nombre de collisions par clé dans la table.

### L’arbre binaire de recherche

L’Arbre Binaire de Recherche ou ABR est une solution alternative à l’utilisation de la table de hachage. En effet l’interface commune de ces deux structures de données nous permet de les substituer facilement l’une l’autre. Il a été décidé de tester cette structure car les ABR permettent une recherche très rapide en log n de par son originalité. En effet, un arbre binaire de recherche est une structure arborescente dont chaque nœud possède au plus deux fils : le fils gauche est inférieur à son père et le droit lui est strictement supérieur.

[ILLUSTRATION]

En gardant cet arbre équilibré, c’est-à-dire en s’assurant que le rang N de l’arbre est complet avant de commencer à remplir le rang N+1, on va donc pouvoir garantir une complexité maximale de recherche en O(log n).

Comme nous l’avons évoqué plus haut l’interface de l’ABR est la même que celle de la table de hachage, nous ne la détaillerons donc pas ici. Toutefois il reste intéressant d’étudier la manière dont nous avons coder la structure arborescente binaire de recherche. En effet, nous nous sommes penchés sur la bibliothèque standard afin de chercher des éléments pouvant accélérer notre travail. Nous nous sommes alors arrêtés sur la classe map. Cette classe générique est d’ordinaire utilisée pour sa capacité à stocker des valeurs par clé mais ce qui nous intéresse est la façon dont elle traite les données et non pas comment elle nous les présente. La classe map utilise en fait une structure d’ABR équilibré pour stocker ses données en mémoire. C’est pourquoi il nous a paru évident d’utiliser la map.

# Outil auxiliaire : génération de phrases

Cette partie du projet sort un peu de la problématique principale. Dans le cadre de cette étude, nous allons analyser les processus aléatoires menant à la création d'une phrase que nous aurons définie.

En effet, le programme va générer aléatoirement une suite de caractères aléatoires dans le but de produire la phrase que nous lui avons donné. Ce processus est très long pour des phrases même courtes. Tout l’intérêt ici étant l’étude du phénomène aléatoire plutôt que l’optimisation temporelle.

## Intérêt

L'analyse des différents temps d'obtention de cette phrase va nous permettre de déterminer un ensemble de statistiques sur les problématiques de génération aléatoires de langages.

## Le programme

### Présentation

Ce programme prend en paramètre une phrase à tester. On va ensuite attendre la fin de N réplications de l’opération de production aléatoire de cette phrase. La probabilité de produire cette phrase est égale au nombre de caractères dans la phrase à la puissance nombre de caractères dans l’alphabet : autant dire que cela peut être énormément long.

Concernant l’aspect aléatoire de ce programme nous avons choisi le générateur pseudo aléatoire Mersenne Twister que nous avions préalablement étudié et que nous pouvons considérer comme performant.

### Utilisation

## Élaboration du programme

### Principe

### Code

# Conclusion

Bilan :

* Problèmes sur l’encodage -> utf8 pour les caractères spéciaux
* Template pour plus de facilité et d’effcacité -> portabilité