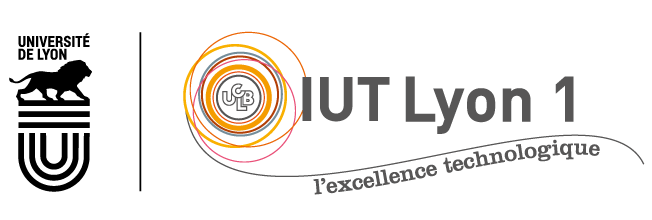
Jérémy LAURENT  
Gérome FERRAND  
Juliette VATON  
Nathan MOUSSU  
Alexandre DUFOUR

COMPTE RENDU DE PROJET

TEXT-MINING

**Enseignant tuteur :** Vincent Vidal

**Semestre**

**INTRODUCTION**

L’année spéciale de DUT INFORMATIQUE est rythmée par le projet tuteuré. Ici, il s’agit ici de développer des programmes qui seront capables de déterminer les notions clés d’un texte. Ces données une fois déterminées, permettront d’assurer plusieurs fonctionnalités.

Le projet étant à l’initiative des étudiants, la première étape a été de la présenter afin de trouver un professeur tuteur. Celle-ci a pour objectif de définir le projet et ses objectifs de manière claire et concise. Vous trouverez alors dans ce rapport le cahier des charges, ainsi que la marge engagée pour concrétiser ce futur projet.

Bonne lecture.

**REMERCIEMENTS**

Vincent Vidal

Nous vous remercions de rendre ce projet possible, de le suivre, de l’évaluer, de nous aider à le réaliser, de l’intérêt que vous lui portez.

Merci pour votre soutien.

Equipe pédagogique de l’IUT informatique

Nous tenons également à remercier l’ensemble de l’équipe pédagogique pour l’enseignement qu’elle nous fournit, mais aussi les conseils qui nous ont permis de nous former en autonomie.

**SOMMAIRE**

[1) DÉFINITION DU PROJET 1](#_Toc532924777)

[A) Contexte 1](#_Toc532924778)

[B) Présentation du projet 2](#_Toc532924779)

[2) ORGANISATION ET GESTION DU PROJET 3](#_Toc532924780)

[A) Répartitions des rôles 3](#_Toc532924781)

[B) Les outils de gestion de projet 3](#_Toc532924782)

[C) Calendrier prévisionnel 4](#_Toc532924783)

[3) PARTIE TECHNIQUE 5](#_Toc532924784)

[A) Technologies utilisées 5](#_Toc532924785)

[B) Structure du programme 7](#_Toc532924786)

[C) Corpus de textes 8](#_Toc532924787)

[D) Notre modèle de traitement : « Bag-of-words » 10](#_Toc532924788)

[E) Gestion des risques 11](#_Toc532924789)

[4) BILAN MI-PARCOURS 12](#_Toc532924790)

[A) Bilan technique 12](#_Toc532924791)

[B) Bilan personnel 12](#_Toc532924792)

[C) Conclusion 12](#_Toc532924793)

**TABLE DES ABRÉVIATIONS**

|  |  |
| --- | --- |
| **BSD** | : Berkeley Software Distribution (licence de distribution logiciel) |
| **CDC** | : Cahier des Charges |
| **C/RNN** | : Convolutional/Recurrent Neural Network, réseau de neurones convolutif/récurrent |
| **EDI** | : Environnement de développement intégré |
| **ERP** | : Entreprise Ressources Planning |
| **IDE** | : Integrated Development Environment, EDI en français |
| **IDF** | : Inverse Document Frequency |
| **IoT** | : Internet of Things |
| **INRIA** | : Institut National de la Recherche en Informatique et en Automatique |
| **IT** | : Information Technologie(s), TIC en français |
| **KNN** | : K-Nearest Neighbors, K-Plus proches voisins en français |
| **NLP** | : Natural Language Processing, équivalent de TALN en français |
| **NLTK** | : Natural Language ToolKit |
| **TALN** | : Traitement Automatique du Langage Naturel |
| **TF** | : Terms Frequency |
| **TIC** | : Technologie(s) de l’Information et de la Communication |

# **DÉFINITION DU PROJET**

## Contexte

Avec l’avènement des ordinateurs personnels et la démocratisation d’internet, les particuliers, les entreprises et maintenant les objets (grâce à l’IoT) émettent toutes sortes d’informations sur les réseaux : photos, vidéos, sons, logs et sous bien d’autres formes encore. Or, 80% de ces ressources se présente dans un format textuel. A titre d’exemple, d’après le site planetoscope.com, en 2015, c’est près de 204 milliards de mails envoyés par jour (hors spams), 200 millions de tweets chaque jour, 65 000 recherches faites sur google chaque seconde et enfin 29 000 Go/s de données publiées dans le monde.

Quantité d’information que de nombreuses sociétés ont l’intention de traiter afin de pouvoir en dégager un avantage concurrentiel certain, et cela au prix de nombreux défis technologiques. Conformément à la loi de Moore, la hausse de performance accompagnée d’une baisse des prix des ressources IT *(e.g.* processeurs et baies de stockage) ont permis le renforcement des datacenters et donc l’ascension du phénomène « Big Data ». En parallèle de ces innovations, les géants du web ont dû investir pour développer de nouvelles méthodes et algorithmes pour traiter ce volume d’information toujours grandissant. Le patron d’architecture MapReduce de Google en est l’une des illustrations ; il permet notamment de parcourir les sites disponibles sur le web dans le but de les indexer, et ce en suivant le modèle Bag-of-Words (*Cf*. III.d. Modèle Bag-of-Words).

Notre projet s’inscrit dans la continuité de cette analyse, appelée Text Mining (*i.e.* fouille de texte en français). Ce domaine de recherche s’appuie sur les techniques du Traitement Automatique du Langage Naturel (TALN ou NLP pour Natural Language Processing), que nous décrirons plus tard dans ce rapport. Cette méthode vise à extraire de manière automatique des informations à partir de documents et répond à de nombreuses applications : la recherche d’informations, les agents conversationnels, l’analyse de comportement, le filtrage des communications ou encore les résumés automatiques de textes. Cependant, il peut être compliqué de procéder à une étude plus approfondie de ce type de format.

Effectivement, d’après Marti A. Hearst (Pr à l’université de Berkeley), « *Le texte exprime une gamme vaste et riche d’information, mais encode cette information dans une forme qui est difficile à déchiffrer automatiquement* ». La difficulté qu’évoque Mme Hearst réside dans le fait que les textes sont soumis d’une part à la complexité syntaxique et d’autre part sémantique. La première étant *a priori* plus simple à traiter automatiquement que la seconde. En effet, il est particulièrement difficile de gérer informatiquement les expressions, les émotions, les variations du langage ou encore saisir le contexte dans lequel le document se place. Telles sont les limites de cette discipline.

## Présentation du projet

Lors de la création du groupe de travail, nous souhaitions nous orienter vers un projet de Machine Learning. Au-delà des applications évoquées en introduction, il s’agit d’un sujet très sollicité sur le marché et qui est synonyme de bouleversements socio-économique non-négligeables. C’était donc une opportunité pour nous de développer notre connaissance dans ce domaine.

Dans un premier temps, nous envisagions d’appliquer notre projet à la reconnaissance d’image (en particulier la reconnaissance de chiffres manuscrits). Effectivement ce domaine d’application (initié par Yann Le Cun dans les années 80) est l’un des plus documenté sur la toile. Nous pensions donc que cela serait une bonne introduction vers ce vaste champ qu’est celui de l’Intelligence Artificielle.

Cependant, la reconnaissance d’images nécessite un traitement des entrées pouvant être long et complexe. En nous orientant d’avantages vers la fouille de texte, nous prévoyons de pouvoir nous focaliser davantage sur le développement et l’optimisation de nos algorithmes. Par ailleurs, au-delà du fait que la plupart des données contenues sur le web sont du texte, nous avons pensé que cette orientation nous permettrait de nous replacer au cœur du traitement de l’information.

Indexation, synthétisation ou encore discrimination de texte (à l’aide de mots-clés), en effet, chacun de nous voit dans le text-mining des applications pouvant nous apporter une plus-value à titre personnel, tant dans la solution que nous proposerons que dans les connaissances que nous acquerrons.

Notre projet s’orientera donc vers **une application opérant sur un corpus de documents**, et sur lesquels diverses requêtes seront appliquées. Voici le plan, avec d’une part les fonctionnalités fondamentales que nous souhaitons mettre en place (voire les détails dans le diagramme de Gantt situé en annexe), et d’autre part, les fonctionnalités secondaires qui seront implémentées si le temps nous le permet :

* Nous commencerons par développer un algorithme que nous nommerons « **noyau** », et dont le but est la **conversion d’un texte en un vecteur déterminant son sens.** Cet algorithme est la base de toutes les autres applications, par conséquent sa fiabilité sera primordiale.
* Nous pourrons ensuite amorcer le développement de la première application, que nous appellerons « **moteur de recherche** ». Sa fonction consiste à **déterminer quels documents répondent le mieux à la requête d’un utilisateur**.
* La seconde application concerne la reconnaissance de document (et plus particulièrement la « **détection de spams** »). L’algorithme sera entrainé à partir d’un data set préexistant, et sera en mesure de **détecter la nature d’un document** avec une précision optimale.
* Notre troisième application correspond à la « **classification** » de documents. Le but de cette dernière et de pouvoir **taguer des documents en fonction de leur thématique**. Nous aurons donc à l’issue de ce programme plusieurs groupes de texte traitant du même sujet.

Dans un second temps, dans le cas où nous parvenons à atteindre les objectifs énoncés au-dessus, nous aspirons à développer d’autres aspects :

* Au terme de ces 3 fonctionnalités, nous envisageons de développer une **interface graphique** permettant de rendre notre solution plus facile d’utilisation et ainsi plus accessible pour des utilisateurs non-initiés à la magie de la programmation.
* Enfin, nous souhaiterions avoir l’opportunité d’**optimiser nos algorithmes** à l’aide de techniques du type **Réseaux de Neurones**.

# ORGANISATION ET GESTION DU PROJET

## Répartitions des rôles

Lors de la création du groupe de travail, il a été décidé de désigner deux chefs de projet afin d’assurer un meilleur suivi. Voici les différents rôles :

* **Lead developer :**
* Jérémy LAURENT
* Gérome FERRAND

Ils coordonnent le projet en programmant les réunions hebdomadaires avec/sans le tuteur pédagogique. Ils réalisent le Gantt et le maintiennent à. Ils veillent également au bon déroulement du projet suivant l’avancement de chaque membre du groupe de travail. Ils participent également au codage.

* **Les développeurs :**
* Juliette VATON
* Nathan MOUSSU
* Alexandre DUFOUR

## Les outils de gestion de projet

Il nous fallait également définir les différents outils de gestion de projet. Il existe des logiciels du style ERP tout-en-un réalisant toutes les tâches souhaitées. Cependant ils ne sont pas gratuits. Nous avons choisi les softwares suivants car nous ne travaillons que sur un seul projet. De plus, notre statut étudiant nous permet d’avoir accès aux licences gratuitement.

**MS Project :** Le Gantt

La création d’un Gantt permet de prévoir l’avancement du projet. Le Gantt est ensuite revu au fur et à mesure afin d’arriver au Gantt réel. Cet outil est plus destiné aux chefs de projets car sa lecture est complexe et offre une vision plus grande que le macro-planning (cf. annexe 1).

**Excel :** Les fiches d’heures



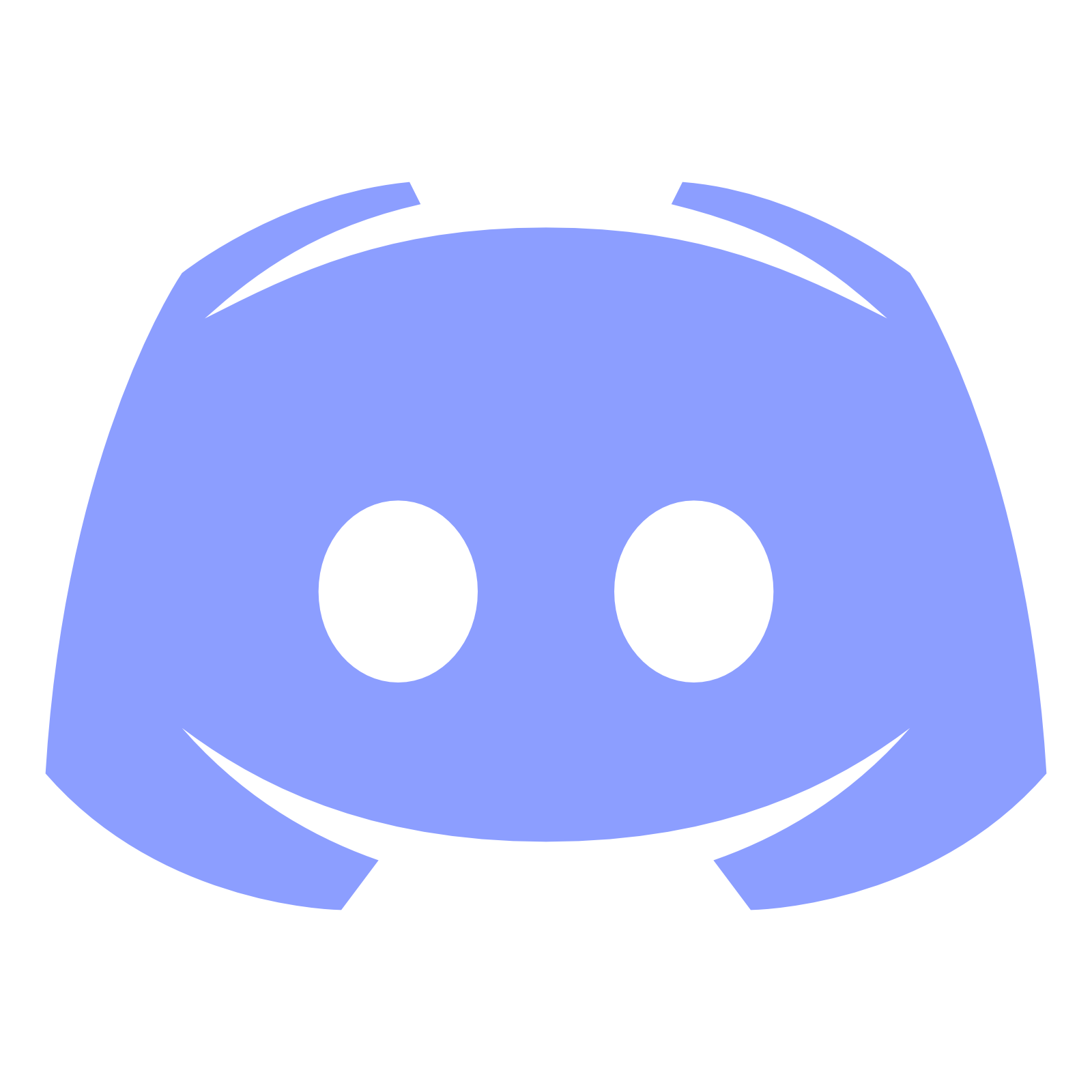
La fiche d’heure va de pair avec le planning car elle permet d’évaluer avec précision la durée de chaque tâche. Par exemple, si nous prenons la semaine 44, il s’agit de la tâche formation sur le Machine Learning. En visualisant la fiche d’heure et le Gantt, nous pouvons voir que cette tâche a duré approximativement 12h cette semaine-là.

**Excel :** Le macro-planning



Le macro-planning permet de suivre les actions hebdomadaires à effectuer. Il est plus simple que le Gantt et permet de visualiser les tâches à réaliser par ressources (membres du projet). Il permet également d’être modifiable rapidement et facilement. Il est plus destiné à être utilisé par l’ensemble du groupe de projet.

**Discord :** Messagerie instantanée / Classage de liens hypertextes



Cet outil permet de communiquer à distance ainsi que d’épingler des messages. Très utile pour retrouver rapidement un lien. Il était également possible d’utiliser un autre logiciel comme « Slack », mais nous connaissions mieux Discord.

**Messenger :** Messagerie instantanée

Cet outil de communication instantanée permet de communiquer rapidement à court terme. Bien qu’il existe des outils spécifiquement dédiés pour les projets, il s’avère efficace et fiable car omniprésent sur tous les smartphones et consulté fréquemment.

## Calendrier prévisionnel

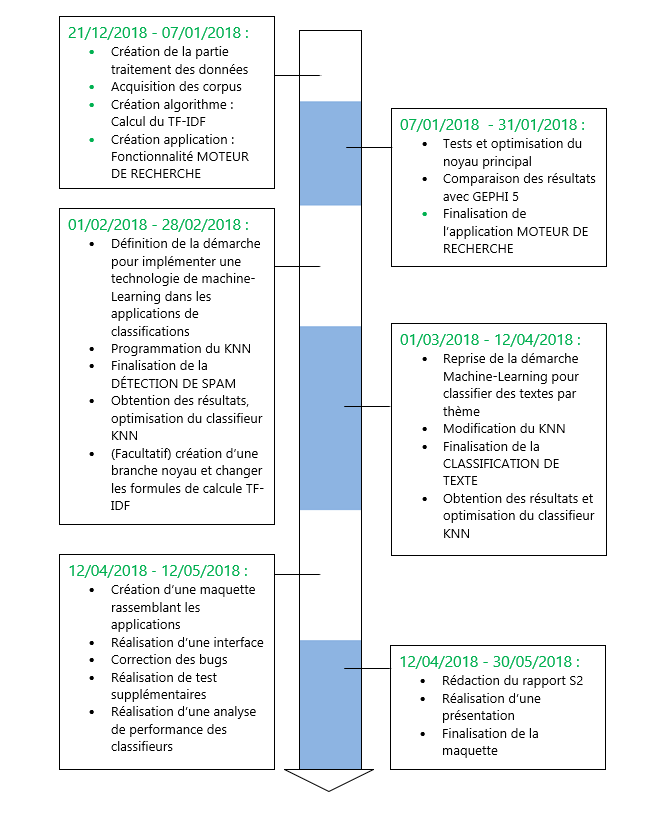


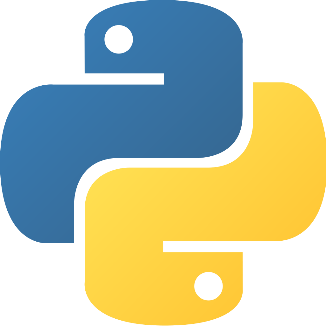
Figure 1: Planning Prévisionnel

# PARTIE TECHNIQUE

## Technologies utilisées

Au cours de nos recherches et nos essaies, nous avons pu aborder de nombreux outils et technologies pour résoudre des problématiques de Machine Learning. Parmi l’ensemble des propositions qui nous étaient faites, nous avons dû choisir celles qui nous paraissaient les plus pertinentes. Néanmoins, nous nous réservons le droit de réajuster nos choix en fonction des difficultés que nous rencontrerons au cours du projet.

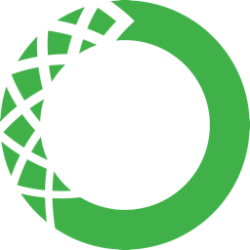
**Langages de programmation** :

**Python** est avant tout un langage de programmation interprété placé sous licence libre (comme la plupart des outils que nous utiliserons pour le développement de notre application). Disponible sur la plupart des systèmes d’exploitation, Python s’est imposé comme la *lingua franca* pour une partie des applications scientifiques, et plus particulièrement dans le domaine de la *data*. Il s’agit d’un langage à la fois accessible pour les néophytes, et puissant grâce à toutes les librairies disponibles ; permettant entre autres la visualisation des données, les calculs statistiques, le travail sur les images, et bien d’autres encore… Par ailleurs, Notons que python est disponibles sur *Jupyter* *Notebook*.

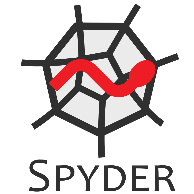
**Python 2 vs Python 3 ?**

Il semblerait que ce langage subisse une phase de transition entre ses 2 variantes. L’arrivée de la nouvelle version implique de nombreuses modifications par rapport à son prédécesseur, rendant certaines parties de code écrit en V2 non-compatible avec la V3. Par conséquent, étant donné que nous commençons un nouveau projet et que nous ne possédons pas d’éléments issues de Python 2, nous travaillerons sous Python 3.

**Outils** :

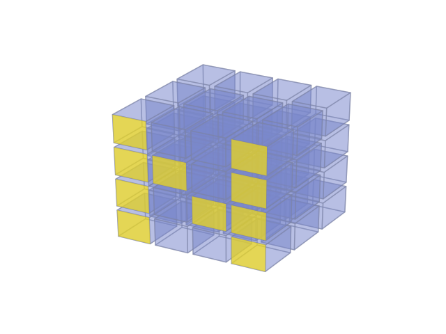
**Anaconda** est un software sous licence BSD (libre) permettant le traitement des données, l’analyse prédictive et autres calculs scientifiques. D’après de nombreuses sources, il semblerait que cette distribution soit à la fois la plus optimisée à l’usage de *Scikit-Learn* (notamment grâce à la bibliothèque *Intel MKL*), mais aussi une (voire la) plateforme la plus populaire au sein de la communauté de la *data science*.

**Jupyter** est un environnement interactif qui permet d’exécuter du code à travers la fenêtre d’un navigateur web. Au même titre qu’Anaconda, il semblerait que cet outil soit largement utilisé parmi les *data scientists* pour l’analyse de donnée. Cette application web est manipulable à l’aide de nombreux langages de programmation comme *JUlia*, *PYThon* ou encore le *R* (d’où son nom). Il est fourni avec la distribution Anaconda, et nous nous sommes déjà familiarisés avec cette solution.

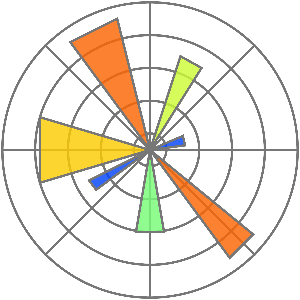
**Spyder** est un EDI (Environnement de Développement intégré, IDE en anglais) qui nous permettra de compiler notre code, même si nous préfèrerons l’utilisation de Jupyter. Fournit avec la distribution Anaconda, cet IDE intègre plusieurs librairies tel que Numpy, SciPy et Matplotlib. Par ailleurs, Spyder est conforme à notre souhait de n’utiliser que des outils libres et open-source.

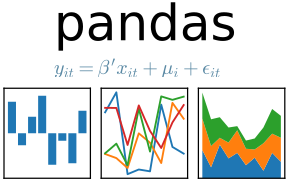
**Bibliothèques** :

**Scikit-Learn** est une bibliothèque libre Python spécialisée dans l’apprentissage automatique. Plus de 1000 personnes contribuent à son développement, notamment des instituts français d’enseignement supérieur (Comme l’INRIA ou encore Telecom Paris Tech). Au centre de notre projet, elle nous permettra de disposer de nombreux algorithmes, accompagnés d’une documentation bien fournie à leur sujet. Cette librairie dépend des packages *NumPy* et *SciPy*.

**NumPy** est une bibliothèque logicielle libre. Il s’agit de l’une des librairies fondamentales de Python. Elle est importée dans la quasi-totalité des projets qui concernent le Machine Learning, et plus généralement, les projets scientifiques. Ses principales fonctions permettent les calculs sur les vecteurs, les tableaux multi-dimensionnels (avec valeurs de même type), opérations d’algèbre linéaire, transformations de Fourier, ou encore la génération de nombres pseudo-aléatoires. *NumPy* recense plus de 600 contributeurs.

**SciPy**, Placé sous licence libre, cette librairie est une version un peu plus évoluée que *Numpy*. Elle contient plus de fonctionnalités que cette dernière, notamment sur des outils d’algèbre linéaire, de distribution statistique, d’optimisation de fonction, etc… *SciPy* nous permettra surtout d’utiliser des *sparses* *matrix* (*i.e.* matrices creuses), correspondant à des matrices contenant essentiellement des 0, transformées pour ne conserver que les valeurs non nulles. Tout comme *NumPy*, *SciPy* dispose de près de 600 contributeurs qui améliorent de jour en jour cette bibliothèque.

**Matplotlib**. De licence libre, *Matplotlib* est la librairie de tracé scientifique principale pour Python. Une bonne façon de représenter nos données, tant en entrée qu’en sortie sous forme de courbes, diagrammes, etc… Combinée avec *Jupyter* *Notebook*, cette bibliothèque nous permettra de former des graphiques directement sur notre environnement par défaut. On peut également exporter les rendus générés sous de nombreux formats (tel que .*jpg*, .*png*, .*pdf*, .*svg*, etc..). C’est grâce à près de 700 personnes que *Matplotlib* continue de se développer.

**Pandas**. La principale fonction de *pandas* est la manipulation et l’exploration des données. Les data se présentent sous la forme d’un *dataframe*/tableau (contrairement à la librairie *NumPy*, les tableaux de *Pandas* autorisent les valeurs de types différents). De plus cette librairie permet d’effectuer des requêtes tout comme le SQL. Enfin, elle gère de nombreux formats de fichier (.*xls*, .*csv*, etc..). C’est près de 1000 personnes qui permettent l’amélioration de *Pandas*.

**NLTK.** Correspondant à « Natural Language ToolKit », il s’agit d’une boîte à outil fonctionnant sous Python comprenant entre autres une librairie servant à réaliser des applications de NLP et un corpus conséquent de textes en diverses langues dont bien sûr l’anglais. Elle nous a été introduite lors des formations et tutoriels suivis, et c’est donc naturellement vers cette solution que l’on s’est tourné afin de réaliser des premiers essais de mise en forme de texte suivant le modèle *Bag-of-words* (voir exemples partie 3.c). Elle possède aussi l’avantage d’être déjà implémentée dans Anaconda.

**Outils de gestion de version** :

Concernant le management du code de notre application, nos principales contraintes sont un service gratuit, une disponibilité multiplateforme et une interface graphique claire. Parmi les choix qui se sont offerts à nous, nous avons choisis les solutions suivantes :

**Git** : Utilisé par près de 12 Millions de personnes, Git est un VCS (Version Control System) permettant entre autres de gérer les différentes versions du projet mais aussi de réunir au mieux les travaux de chacun d’entre nous. L’un des principaux avantages de l’outil de versioning mis au point par Linus Torvalds, est qu’il dispose d’un modèle distribué (permettant notamment le travail hors-connexion). Git se démarque de SVN ou Mercurial par sa rapidité d’exécution ou encore le travail en branche. Par ailleurs, notons que c’est qui nous a été présenté en début d’année, nous avons donc déjà eu l’occasion de l’appréhender.

**GitKraken** : est l’environnement graphique utilisant Git et qui répond au mieux à nos critères. Malgré sa licence propriétaire (axosoft) et un code inaccessible, il permet le dépôt de nos projets sur plusieurs serveurs. En plus d’une représentation des versions limpide, GitKraken contribuera à notre gestion de projet grâce à certaines de ses caractéristiques tel que le suivi des date limites, les mentions à d’autres membres du projet ou encore la mise en place de « checklists ».

**GitLab CE** : Afin que chacun de nous puisse travailler simultanément nous avons fait le choix d’exploiter un serveur distant et en particulier ceux fournis par GitLab. Leur édition communautaire est la seule open-source parmi leurs concurrents (GitLab EE, GitHub, BitBucket et Coding), mais aussi celle conseillée par notre tuteur. Néanmoins, l’édition « community » ne dispose pas de cloud ; si nous souhaitons être indépendamment de la connexion à un serveur personnel, nous pourrons éventuellement passer sur l’édition « Entreprise » qui propose un hébergement de nos ressources.

## Structure du programme

Figure 2: Pipeline du programme

## Corpus de textes

Dans le cadre de ce projet, nous entendrons par texte une suite de phrases. Ces phrases sont définies comme une suite de caractères commençant par une majuscule et se terminant par un des caractères suivants : “. ! ? “. Nous exclurons, pour notre corpus, la présence d’images ou de caractères spéciaux, et ne traiterons que de textes libres (format .txt).

Concernant le format d’encodage de nos fichiers textes, nous favoriserons l’*utf-8*, qui a l’avantage de supporter virtuellement n’importe quel langage naturel, de permettre l’usage d’accents et d’être très largement utilisé. De plus, l’utf-8 a l’avantage d’être toujours compatible avec le code ASCII, car les deux sont codés sur 8 octets, et l’utilisation d’une autre représentation d’Unicode telle que utf-16 constituerait un “gaspillage” de mémoire, puisque l’on utiliserait seulement huit des seize attribués en mémoire. Nous avons choisi l’anglais comme langue de travail, d’une part par la plus faible proportion d’homonymes par rapport au français, mais aussi pour des raisons de compatibilité et de cohérence dans l’optique de l’utilisation de datasets (notamment pour les applications dérivées de notre algorithme) qui sont pour la majorité en anglais. Par ailleurs, nous avons opté pour la définition d’une taille minimum pour les textes exploités. Ainsi, afin d’assurer une meilleure normalisation des données et une représentation plus cohérente avec la réalité, nous avons décidé, à l’heure actuelle, de limiter les documents à un minimum de dix phrases : la raison de ce minimum est expliquée plus loin dans cette section.

Le corpus, du fait d’une utilisation orientée Machine Learning, devra comporter des textes représentatifs qui constitueront le training-set, ainsi que d’autres qui constitueront le data-set. Il faut définir des training-sets spécifiques pour chaque type d’utilisation : en effet les caractéristiques recherchées varieront en fonction de l’application cible.

Il faut considérer les textes comme des séries statistiques dont les observations sont les mots qui les composent, et le sens des textes une interprétation de ces observations. Ainsi, partant de ce principe il y a plusieurs règles à respecter :

- Un échantillon doit avoir une taille minimale afin de pouvoir établir des approximations pertinentes. Pour établir un training-set satisfaisant, c’est à dire un groupement de texte dont on sait la signification, nous avons choisi de reprendre la formule donnée par Wikipedia et qui est la plus communément admise au sein des instituts de statistique :

* n : Taille d'échantillon minimale pour l'obtention de résultats significatifs pour un événement et un niveau de risque fixé
* t : Niveau de confiance (la valeur type du niveau de confiance de 95 % sera 1,96)
* p : proportion de mots qui donnent son sens au texte
* m : Marge d'erreur (généralement fixée à 5 %)

**Comment obtenir « p » ?** C’est en analysant le texte et en établissant des groupements sémantiques de mots que nous parviendrons à fixer la proportion de mots significatifs (i.e. qui donnent leur sens au texte) et ainsi à définir la taille minimale d’un texte de training-set. Pour ce faire, nous allons analyser manuellement le nombre de mots qui définissent un texte et allons répéter cette opération sur 25 textes différents. La moyenne obtenue sera « p » (cf. annexe).

Il s’agit en partie de choix arbitraires dans la mesure où le langage est un phénomène naturel qu’il est difficile d’approximer statistiquement.

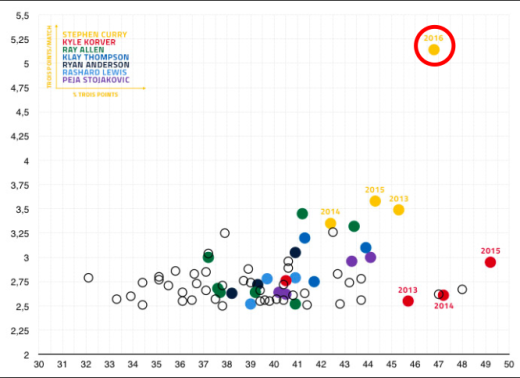
**

Figure 3: L’exemple de la performance hors-normes d’un joueur de Basketball.

Il faut nettoyer l’échantillon avant de l’exploiter réellement. Il y a bien sûr l’élimination des mots de liaison, mais des problèmes plus complexes se posent dans le cas de ce qui serait une aberration dans une série statistique. Que faire d’un mot isolé au sein d’un texte ? Cela signifie-t-il qu’il est particulièrement important ? Ou l’inverse ?

Ensuite, en partant du principe que le corpus d’entraînement est représentatif du reste des textes du même thème (i.e. que nos statistiques sont fiables), il s’agira d’exploiter ces données statistiques à l’aide d’outils qui mettront en évidence les points communs et les différences des textes entre eux ou bien les éléments caractéristiques au sein d’un texte.

Par exemple, réaliser une régression linéaire de la probabilité qu’un texte soit un spam en fonction du nombre d'occurrences d’un certain mot ou groupe de mots. Le langage étant une donnée complexe, il est possible d’étoffer cette analyse à l’aide de régressions à plusieurs dimensions : on pourrait ainsi définir le sens d’un texte à partir de plusieurs facteurs indépendants entre eux.

Pour résumer, l’enjeu consiste à quantifier correctement un texte afin de constituer soit des vecteurs qui représentent un groupe de mots pondéré, soit des catégories de mots. À partir de là, il est possible d’établir des distances entre les différents vecteurs pour extraire du sens.

Enfin, nous privilégierons des textes au contenu spécifique à un domaine particulier pour chaque dataset/corpus ; en effet, notre algorithme noyau (bag of words) se prête parfaitement à des applications dans un contexte précis, mais présente des limites lorsqu'il est confronté à des texte appartenant à des domaines trop éloignés, car il n’établit pas de lien sémantique entre les mots et les considère comme entièrement indépendants les uns des autres (notion de one-hot-encoded vector, expliquée plus loin). Traiter des textes trop différents sémantiquement signifierait un potentiel d’erreur trop important, bien que nous travaillions d’ores et déjà sur une méthode en parallèle, telle que le *word embedding (plongement de mots)* qui établirait des liens sémantiques (par exemple en regroupant les synonymes).

Les données de type textuel sont des données brutes et ne présentent pas de “features” exploitables directement : c’est le rôle de notre “noyau” de rendre le texte exploitable. Elles demandent donc un prétraitement rigoureux et ciblé. Avant de pouvoir exploiter ces données textuelles, il faut donc passer par un certain nombre d’étapes :

* La récupération du corpus se fait par scraping (notamment via les bibliothèques scrapy ou beautiful sous sur Python) ou en téléchargeant des fichiers textes. Cela peut demander l'utilisation de l’outil RegEx (Regular Expression) afin de récupérer uniquement les parties souhaitées.

## Notre modèle de traitement : « Bag-of-words »

C’est d’un modèle de représentation de texte utilisé pour des applications de NLP (Natural Language Processing, exemples : classification de documents, détection de spam) ou de recherche d’information. **Il s’agit de représenter le texte par un dictionnaire de mots-clés appelés *tokens* auxquels on donne un score basé sur la fréquence d’apparition de ce mot-clé dans le texte et l’ensemble du corpus.**

L’obtention de ce dictionnaire se fait à travers plusieurs étapes :

**Tokenisation :**

C’est le découpage du texte, sous la forme d’une chaîne de caractère unique, en un tableau de mots-clés. Par exemple, *tokeniser* la phrase “The bag-of-words model is a simplifying representation used in natural language processing and information retrieval.” donnera le tableau:

['The', 'bag-of-words', 'model', 'is', 'a', 'simplifying', 'representation', 'used', 'in', 'natural', 'language', 'processing', 'and', 'information', 'retrieval', '.']

Mais la tokenisation seule présente des inconvénients : elle n’essentialise pas assez le texte en prenant en compte des détails qui ont un sens au niveau local, mais pas lorsqu’on cherche la récurrence d’un concept, comme la ponctuation, les majuscules, la conjugaison, les accords de genre et de nombre et les mots de liaison. On procède donc à d’autres étapes avant d’utiliser les données afin de les normaliser.

**Stop-Word :**

Les stop-words sont un ensemble de tokens dont on juge qu’ils ne sont pas dignes d’être conservés, ce sont ceux faibles sémantiquement, on les supprime alors simplement du dictionnaire de tokens. Cela pourrait donner pour notre exemple le tableau :

['The', 'bag-of-words', 'model', 'simplifying', 'representation', 'used', 'natural', 'language', 'processing', 'information', 'retrieval', '.'] On a alors supprimé les tokens ‘is’, ‘a’, ‘in’, et ‘and’. Le choix des stop-words n’est pas absolu et peut dépendre d’une méthode à une autre.

**Stemming (racinisation ou stemmatisation) :**

C’est une technique de normalisation des données. Il s’agit de réduire les tokens à un radical en supprimant les préfixes et suffixes potentiels. Tous les mots d’une même famille sont alors représentés par un seul token qui est leur radical, qui en général n’est pas un mot existant.

Pour notre exemple on pourrait obtenir comme tableau après stemming :

['the','bag-of-word', 'model', 'simplifi', 'represent', 'use', 'natur', 'languag', 'process', 'inform', 'retriev', '.']

Il existe plusieurs algorithmes de stemming, Porter, Lancaster ou bien Snowball par exemple.

**Lemming (lemmatisation) :**

C’est une autre technique de normalisation des données. Il s’agit ici de regrouper les tokens de même famille sous une entité appelé lemme qui est la forme canonique de cette famille. Cette forme peut être le verbe à l’infinitif pour les verbes, ou le mot au masculin singulier pour les autres types de mots. Il s’agit donc de supprimer toute conjugaison ou accord en genre et en nombre.

Il faut au préalable réaliser un POS-Tagging (Part-Of-Speech Tagging), c’est-à-dire un étiquetage des tokens avec leur forme, soit par exemple associer à ‘used’ l’étiquette ‘verbe conjugué’, afin que l’algorithme de lemming puisse le traiter correctement.

Pour notre exemple (*non-stemmisé*) cela peut donner après POS-Tagging :

[('The', 'DT'), ('bag-of-words', 'NNS'), ('model', 'NN'), ('simplifying', 'VBG'), ('representation', 'NN'), ('used', 'VBN'), ('natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('information', 'NN'), ('retrieval', 'NN'), ('.', '.')]

Le *Tag* associé à chaque type de mot peut varier d’une méthode de POS-Tagging à une autre, ici on identifie les verbes par le ‘V’ en première lettre de leur étiquette.

On peut alors lemmatiser le résultat, ce qui peut donner :

['The', 'bag-of-words', 'model', 'simplify', 'representation', 'use', 'natural', 'language', 'processing', 'information', 'retrieval', '.']

**Scoring :**

Une fois les données mises sous forme de dictionnaire de tokens, il s’agit à présent d’associer à chaque token un *score* qui représentera son importance dans une texte donné du corpus. Ce score dépend plus ou moins directement de son nombre d’apparition dans ledit texte, c’est le modèle de score le plus naïf.

On peut aussi associer à chaque token sa fréquence d’apparition (TF : Term Frequency), qui est son nombre d’apparition divisé par le nombre total de token dans le document.

On peut enfin calculer grâce aux autres textes du corpus l’IDF (Inverse Document Frequency) du token pour le texte, qui sera un nombre d’autant plus grand que le token apparaît dans peu d’autres textes. **En associant aux tokens TF\*IDF on obtient un score représentatif de l'importance du token dans le texte ainsi que de sa spécificité dans ce texte par rapport aux autres du corpus, ce qui permet de mieux identifier les thèmes propres à chaque texte.**

## Gestion des risques

Concernant l’implémentation technique de l’algorithme, l’un des principaux problèmes que nous avons détectés est le choix des composants du vecteur destiné à quantifier un texte. S’agit-il de catégories de mots (typiquement un champ sémantique), ou bien s’agit-il de mots en eux-mêmes, dont la représentativité serait à même de définir le texte étudié. L’avantage d’établir des catégories de mots serait une analyse plus fine du texte, un groupe de mots étant plus susceptible de définir une notion qu’un mot isolé. Cependant, il est techniquement difficile de constituer des champs sémantiques (le stemming ayant ses limites. Ex : travail & labeur).

Un autre risque que nous avons rencontré concernait plutôt la gestion du projet en lui-même : initialement, le projet devait porter en intégralité sur le thème du Machine-Learning voire du Deep-Learning. Après avoir commencé à réunir des connaissances à ce sujet, nous avons décidé de nous recentrer davantage sur le prétraitement du texte afin de pouvoir mobiliser plus aisément et plus rapidement les connaissances en cours d’acquisition.

A ce sujet, le risque dont il était question était la mise en place d’un projet irréalisable techniquement dans les limites de temps imparties. Nous avons su le prendre en compte suffisamment tôt et nous sommes réorganisés efficacement.

Cela dit, nous gardons toujours en tête qu’il s’agit d’un thème très riche (surtout pour un niveau D.U.T.) et que le besoin d’acquérir en continu des connaissances de manière autodidacte constitue un enjeu constant.

# BILAN MI-PARCOURS

## Bilan technique

Au départ, nous souhaitions travailler sur un projet en rapport avec le machine-learning car Il s’agit d’un domaine en plein essors et qui possède de nombreuses applications : reconnaissance d’image, prise de décision, analyseur, classifieur etc… Nous avons finalement choisi le texte-mining car il répondait à une problématique à laquelle de nombreux domaines sont confrontés : la lecture et compréhension d’un texte.

Du fait de notre faible expérience en informatique la définition de notre projet (les objectifs et les solutions à mettre en œuvre) dura tout le premier semestre. Nous avons également dû nous former aux différentes technologies de machine-learning. Notre projet s’est complexifié notamment à cause des formations que nous avons réalisée, ce qui a conduit à une redéfinition du CDC et de ses solutions.

Nous avons finalement décidé de réaliser un logiciel qui comporte un algorithme principal traitant et nettoyant les données (donc des mots) ainsi que diverses applications telles que : un moteur de recherche, un classifieur de texte, un détecteur de spam. Dans le cadre de notre projet les données seront des textes au format txt.

Nous émettons cependant des réserves sur la réalisation des points suivants

* L’interface utilisateur : il est possible que cette tâche se retrouve hors projet dans le cas où nous aurions du retard.
* Un réseau de neurones pour classifier les textes : si tel était le cas, nous concentrerions nos recherches sur l’optimisation du KNN en jouant avec les paramètres afin d’obtenir les résultats les plus pertinents dans notre classification de textes.

## Bilan personnel

Le projet étant intéressant, la motivation de groupe de projet est présente depuis le début. Bien que le sujet soit complexe, l’ensemble du groupe de projet s’est autoformé et est très impliqué dans le projet (228h au total de travail). Nous réalisons des réunions hebdomadaires afin de faire le point sur l’avancement des différentes tâches. Des comptes rendus y sont rédigés à chaque séance.

## Conclusion

Ce premier semestre nous a permis de proposer des solutions pour créer l’algorithme principal. Le second semestre nous permettra d’acquérir le reste des compétences nécessaire pour développer et mettre en place les applications de classification de texte. Le but final étant de faire intervenir des solutions types machines-learning sur ce type d’application. Afin de gagner du temps, nous allons également utiliser la bibliothèque NLTK qui permet de réaliser le nettoyage ainsi que le traitement des données. Nous allons coder l’algorithme principal pendant les vacances de noël afin d’obtenir des résultats le plus tôt possible. Cela nous permettra de valider la méthode de traitement pour pouvoir passer au codage des applications.

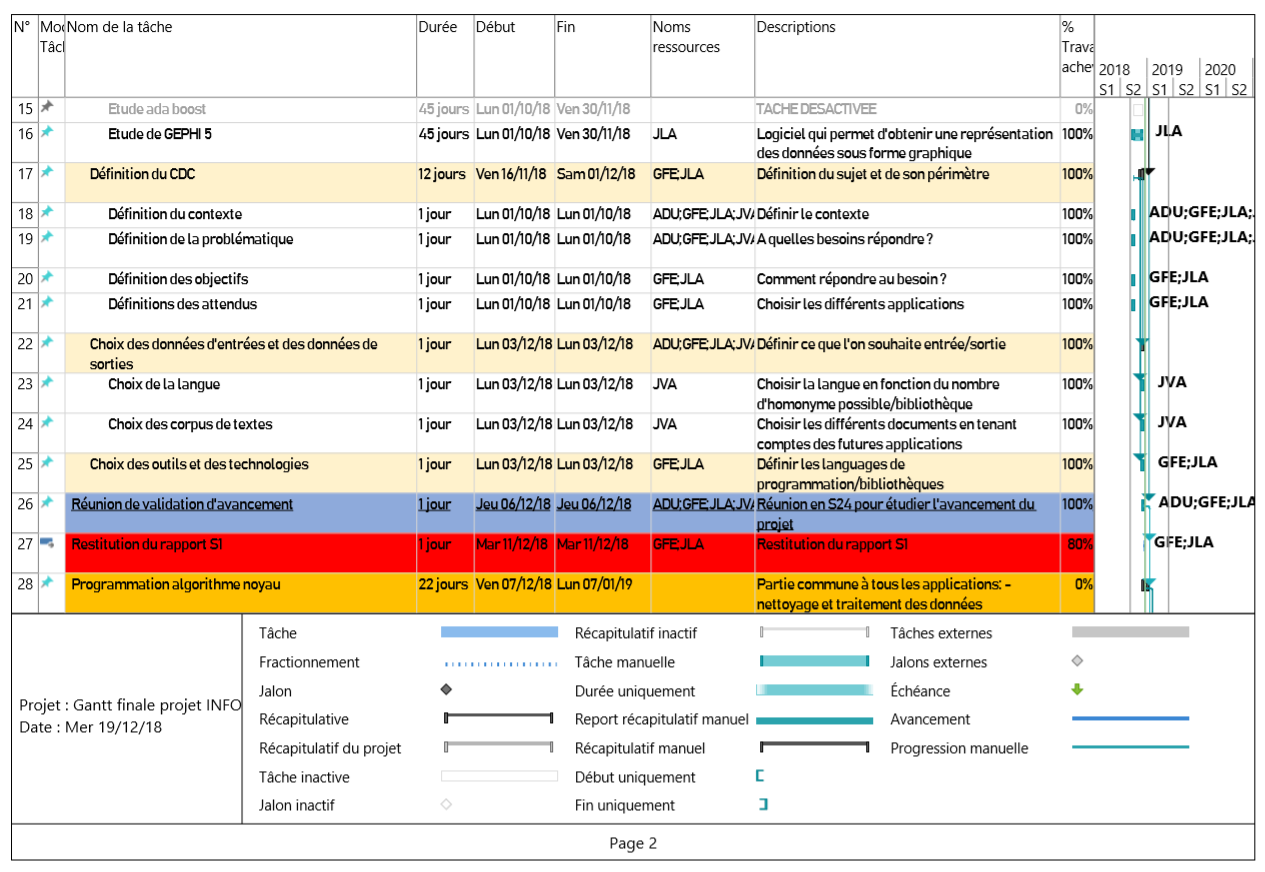
**Annexes**

1. Le planning prévisionnel complet
2. Les fiches d’heures
3. Comment trouver p ?

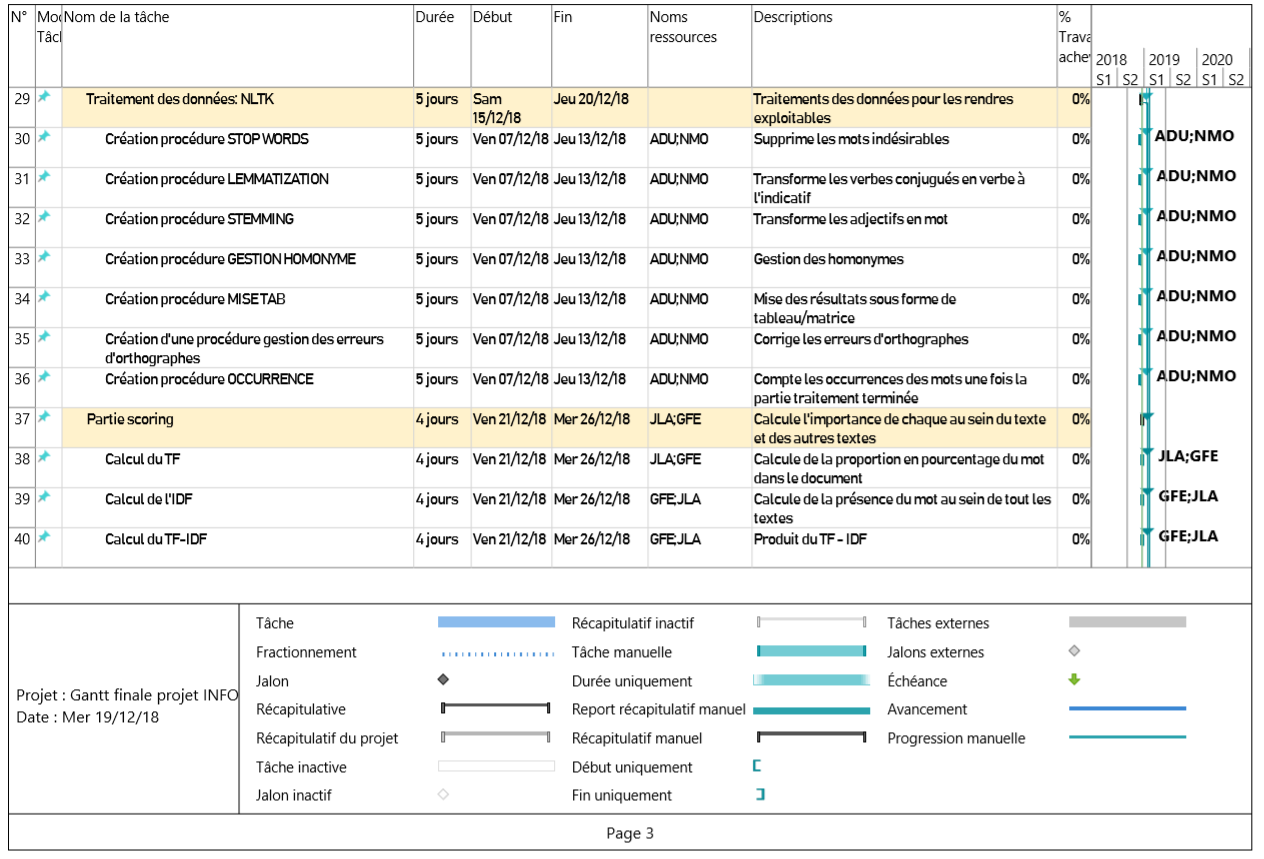
Le planning prévisionnel complet



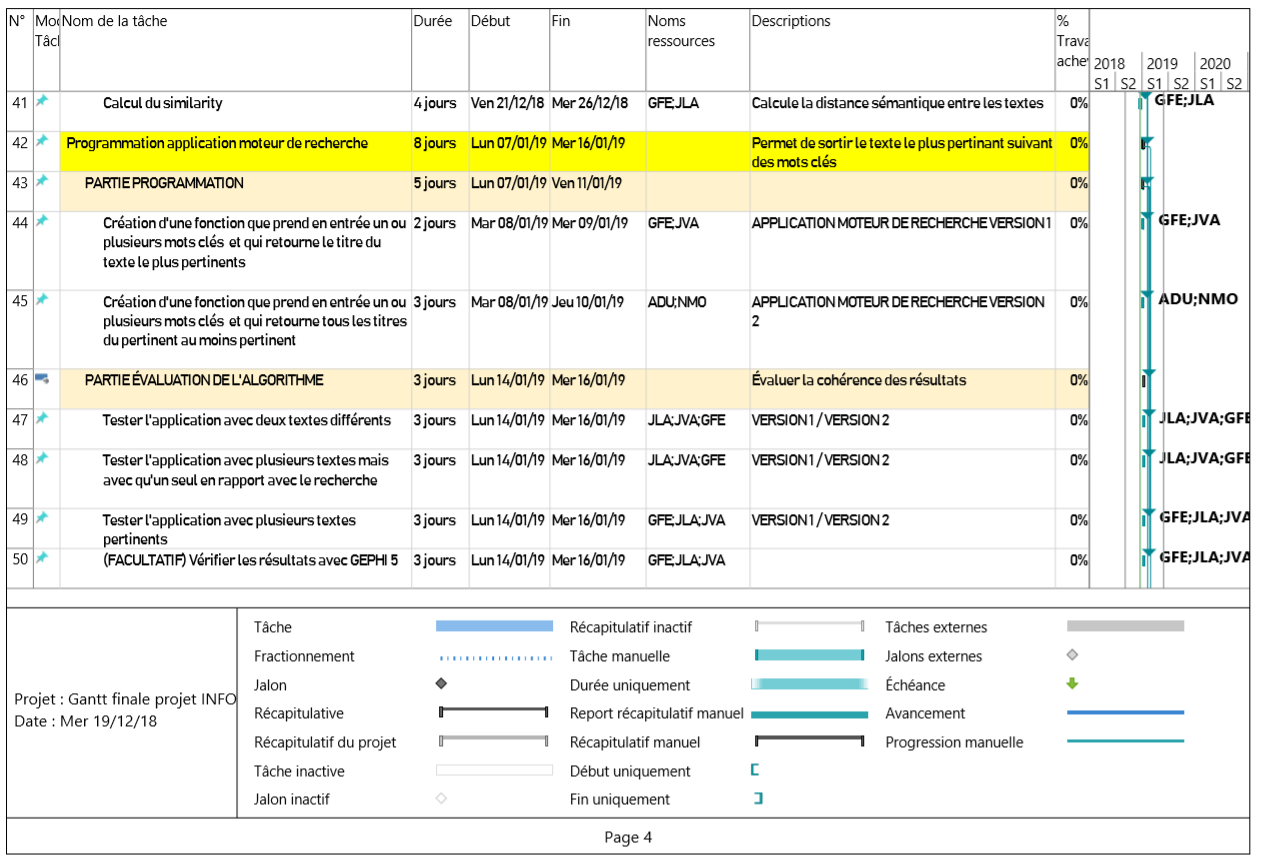
Le planning prévisionnel complet



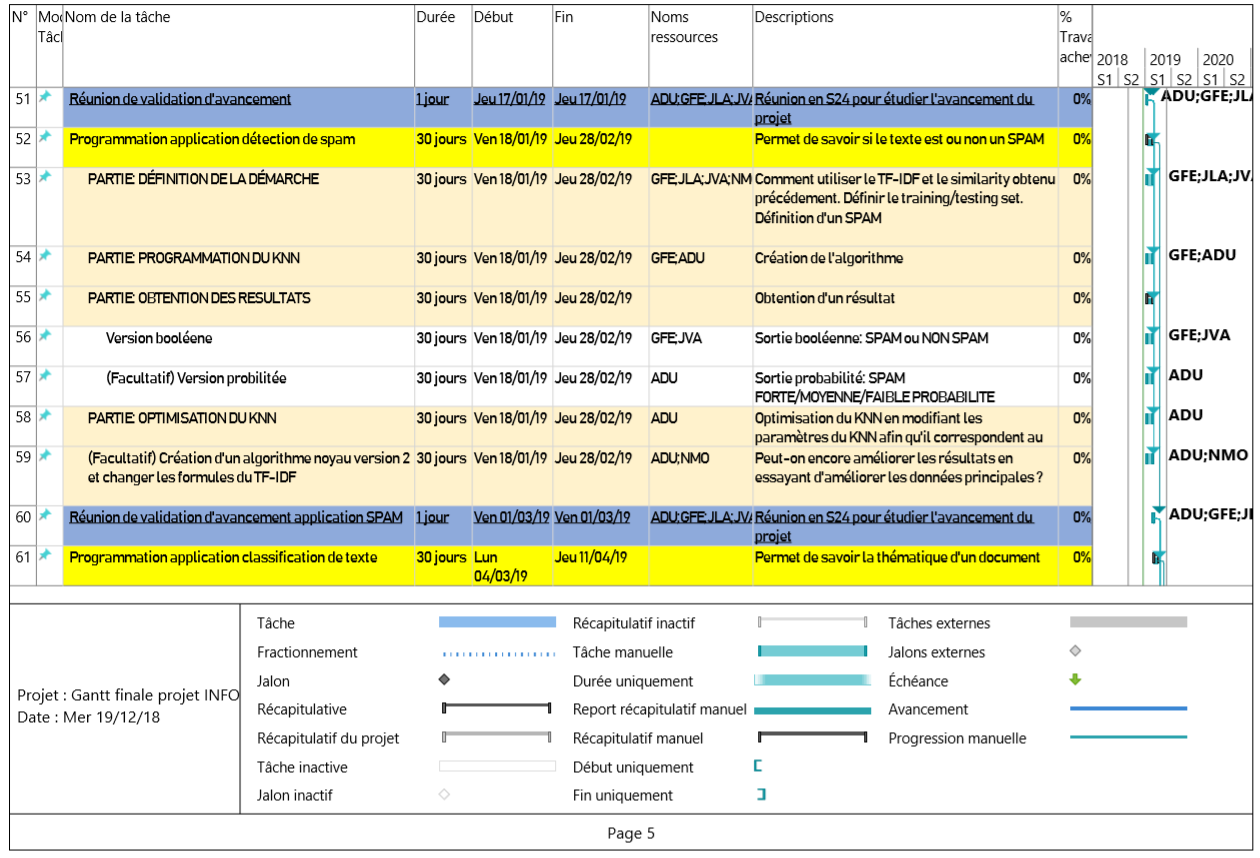
Le planning prévisionnel complet



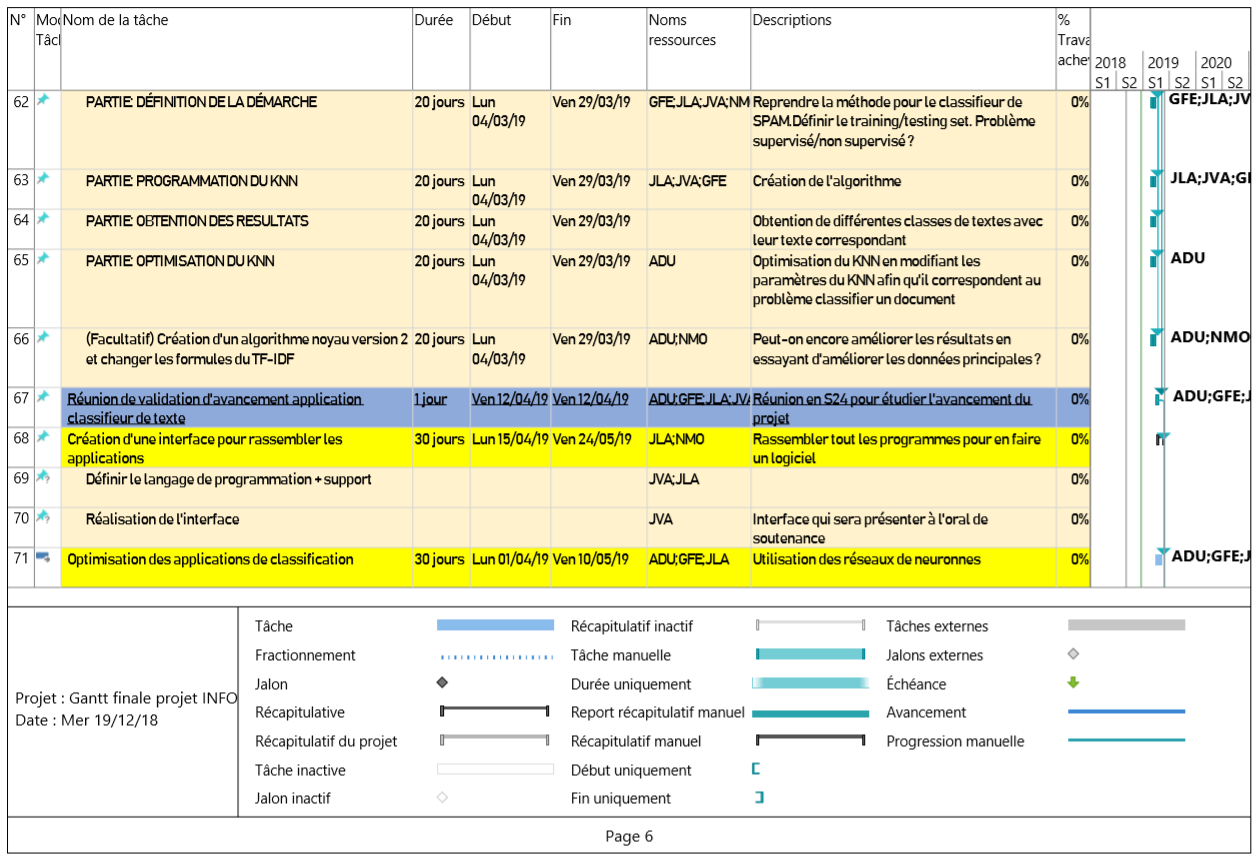
Le planning prévisionnel complet



Le planning prévisionnel complet



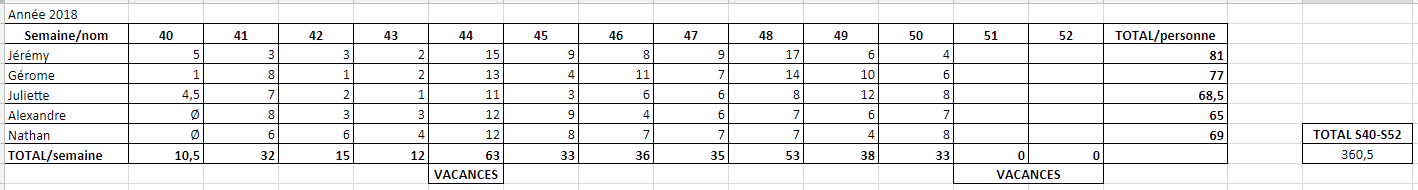
Le planning prévisionnel complet



Le planning prévisionnel complet



La fiche d’heure



Comment trouver p ?

**Comment obtenir « p » ?** C’est en analysant le texte et en établissement des groupements sémantiques de mots que nous parviendrons à fixer la proportion de mots significatifs (i.e. qui donnent leur sens au texte) et ainsi à définir la taille minimale d’un texte de training-set. Pour ce faire, nous allons analyser manuellement le nombre de mots qui définissent un texte et allons répéter cette opération sur 25 textes différents\*. La moyenne obtenue sera « p ». (\*il est admis que c’est à partir de 25 que l’échantillon devient représentatif de l’ensemble des textes.

Cependant, s’il s’avère que ce n’est pas suffisant (par exemple l’écart-type de la série est trop important), nous pourrons appliquer la formule suivante :

Où ε représente la marge d’erreur par rapport à la moyenne observée (arbitrairement fixée à 5%), σ l’écart-type et N le nombre de textes étudiés.)