Compte-rendu du projet
Base de données relationnelle ENRON

Introduction

Initialement, on possède des données sous la forme d'une arborescence de boîtes mails de personnes travaillant dans l'entreprise Enron, et d'un fichier XML contenant des données de certains employés.

À partir de ce jeu de données, nous avons réalisé une application web à destination d'enquêteurs, permettant de visualiser les données intéressantes, grâce à des requêtes. Il s'agit notamment d'informations sur les employés, et sur les mails échangés.

Pour cela, nous avons préalablement construit un modèle de données en utilisant le système de gestion de bases de données postgreSQL, avant de peupler les tables définies dans ce modèle, grâce au langage de programmation Python. La dernière étape consistait à réaliser l'application web avec le framework Django. Celle-ci permet de visualiser les données sous forme de tableaux et de graphiques, suivant les paramètres entrés dans des formulaires. Le type de visualisation diffère ainsi selon les différentes requêtes proposées.

I - Présentation

En ce qui concerne le schéma du modèle de données, nous avons décidé de le créer sous la forme de 9 tables, qui sont les suivantes : *Employee*, *Address*, *Address_Communication*, *Communication*, *Conversation*, *Extern*, *Communication_Extern*, *Content* et *Word*. Nous avons fait ce choix en fonction des données que nous souhaitions récupérer, et pour prévoir les futures visualisations. Pour donner quelques précisions, une communication correspond à l'échange d'un mail entre deux personnes, et une conversation contient plusieurs communications. Ces tables ont des relations entre elles. En effet, un employé possède une ou plusieurs adresses emails, ce qui correspond à une relation 1:n. Cependant, une adresse email est associée à plusieurs communications, et une communication contient plusieurs adresses emails, c'est pourquoi nous avons ici une relation n:n. Celle-ci nous a conduit à construire une table *Address_Communication* permettant de respecter les règles de normalisation. Il en est de même pour la table *Communication_Extern.*

Pour une communication, nous avons défini différents attributs. Tout d'abord, l'attribut receiver regroupe les trois différents types de réception : To, Cc et Bcc. Les colonnes *sender* et *receiver* sont généralement composées des adresses emails. Parfois ce sont des noms de personnes, pour les cas où l'on ne possède pas leurs adresses dans la table *Employee* pour les employés, et dans Extern pour les personnes extérieures à l'entreprise.

Nous avons choisi de définir qu'une conversation correspond à un message id unique, et également à un fichier, c'est-à-dire à l'échange de plusieurs mails ayant un sujet commun.

Nous avons fait le choix de séparer le contenu des mails, des informations des personnes associées à celui-ci (telles que le type de receiver ou le type d'échange), ce qui a conduit aux

tables *Content* et *Communication*. En effet, les regrouper aurait créé une redondance dans le cas où plusieurs *receiver* recevraient le même mail.

Nous avons fait le choix de créer la table Word, pour y stocker les mots clés de chaque conversation. Une ligne de cette table correspond à un mot clé et à son nombre d'occurrences dans la conversation. Cela nous a permis à l'aide d'une requête, de déterminer quelle est la conversation correspondant le plus aux mots fournis par l'utilisateur.

Initialement, nous avions décidé de stocker toutes les personnes dans une même table, mais nous avons finalement ajusté notre modèle de données en créant la table *Extern* afin de séparer les personnes extérieures de celles travaillant dans l'entreprise. En effet, ce n'était pas pertinent de regrouper les employés et les personnes extérieures dans une même table, alors que nous possédons pas certains attributs pour les externes, comme *category* ou *mail_box*. Aussi, cela nous a permis de réaliser plus facilement nos requêtes, notamment celles concernant les attributs d'un employé.

À propos de l'application, nous avons créé différentes pages, chacune correspondant à une requête. Pour ces dernières, nous avons choisi d'ajouter certaines visualisations que nous avons estimées judicieuses, notamment en réalisant un diagramme circulaire pour représenter le pourcentage d'employés et d'externes inclus dans la base de données. De plus, concernant la visualisation des employés ayant envoyé et/ou reçu un certain nombre de mails, nous avons ajouté la possibilité de voir le détail des échanges concernés. Enfin, pour la représentation des jours ayant les plus grands nombres d'échanges de mails, le résultat est affiché sous forme d'un diagramme en bâtons, complété par un tableau classé par ordre décroissant.

II - Difficultés, surprises

Tout au long du projet, nous avons dû faire face à certaines difficultés, notamment pour l'étape de peuplement. Cela était essentiellement dû à la présence de beaucoup de cas particuliers dans les différents mails.
En effet, en prenant l'exemple des dates, différents formats sont existants dans les données. Pour traiter ces différents cas, nous avons créé deux fonctions de traitement des dates, une pour le premier mail d'une conversation, l'autre pour les mails réponses. Pour le cas où la date n'est pas reconnue, nous avons décidé de soit en définir une par défaut, soit de reprendre la dernière date mémorisée.
Aussi, nous avons eu d'autres difficultés, pour récupérer les noms et adresses des personnes communicants dans une conversation, pour la même raison que citée précédemment. Nous avons ainsi défini plusieurs regex, qui s'adaptent à chaque situation, pour éviter un maximum d'erreurs lorsque certains mails ne possèdent pas tous les attributs communs. Pour ce traitement, les différents cas particuliers concernaient le format des lignes de "From", de "To" ou encore de "X-cc", et d'autres encore. En effet, les séparateurs des personnes n'étaient pas toujours les mêmes. Parfois, il y avait des balises entre les noms, avec la présence ou non de virgules ou de points-virgules. Aussi, il arrivait que le prénom et le nom de chaque personne soient aussi séparés, ce qui complexifie encore plus le traitement. De plus, il pouvait y avoir une seule personne ou plusieurs par groupe, ce qui modifiait systématiquement la méthode de traitement. Nous stockons ce traitement dans deux fonctions externes, en fonction de s'il s'agit du premier mail de la conversation ou d'une réponse, et nous y faisons appel à plusieurs reprises dans le script de peuplement de Communication et Conversation.

Une autre difficulté était de déterminer comment séparer les mails des conversations, pour les traiter séparément. Nous avons décidé de les séparer seulement à l'élément " original message" lorsque l'on trouvait dans une conversation, puisqu'il s'agit du cas majoritaire.

Nous avons également eu quelques surprises notamment avec la longueur de certains des mails pour la table Content qui dépassait la capacité maximale. Au lieu d'augmenter le nombre de caractères pour cet attribut, nous avons préféré couper le mail pour ne garder que le début, dès que cette erreur apparaissait.

Par ailleurs, pour éviter d'avoir un trop grand nombre de lignes dans la table Word, nous avons fait le choix d'enlever les ponctuations, les *stopwords* afin d'obtenir une meilleure exploitation des données, pour récupérer uniquement les mots les plus pertinents pour la visualisation.

III - Limites

Le traitement des données et l'application ont certaines limites, et peuvent être améliorées. En effet, le temps d'exécution est assez long pour le peuplement des tables Content et Word, et on estime qu'il est possible que le traitement puisse se réaliser avec d'autres techniques plus rapides et efficaces. De façon générale, il existe probablement d'autres techniques plus simples pour aller récupérer les informations dans les mails.

Concernant les requêtes, nous aurions pu ajouter d'éventuelles visualisations supplémentaires. En effet, nous aurions pu réaliser une requête avec l'attribut *type_receiver* de la table Communication. Il y aurait également été possible de visualiser le nombre moyen de mails dans pour conversation, ou encore déterminer la durée moyenne d'une conversation dans le temps. Cela aurait pu être intéressant de compléter nos visualisations avec plus d'informations concernant les pièces jointes.