Projet Text Mining - Analyse régionale des offres d'emploi

Davy Darankoum, Franck Doronzo, Gwladys Kerhoas

Février 2022

Introduction

Dans le cadre de notre projet en Text Mining, nous devons extraire et traiter un corpus à partir des sites d'offres d'emploi accessibles en ligne et produire une analyse à l'aide d'une application R Shiny. Celle-ci est destinée à guider l'exploration et l'analyse du corpus de manière visuelle et intéractive. Nous avons choisi de nous interésser aux offres d'emploi disponibles sur le site de Pôle Emploi. L'objectif est d'analyser le corps de texte des différentes offres présentes sur le site choisi. Nous avons mis l'accent sur un domaine en particulier, à savoir les offres d'emploi comportant le mot "data".

Nous verrons dans ce rapport la démarche d'extraction des données, la structure de la base de données ainsi que l'architecture de l'application web intéractive. Nous proposerons également une démonstration de l'utilisation de notre application R Shiny et des différentes analyses ainsi que les conclusions que nous avons pu en retirer.

1 Extraction du jeu de données

1.1 Site choisi

Le choix du site utilisé pour récupérer les offres d'emploi s'est porté sur pôle emploi. En effet, celui-ci dispose d'une api très bien documentée, ce qui nous a permis de pouvoir récupérer l'ensemble des offres d'emploi voulu au format JSON sous R.

1.2 Principe utilisé : API

Une API (application programming interface ou interface de programmation d'application) est une interface logicielle qui permet de connecter un logiciel ou un service à un autre logiciel ou service afin d'échanger des données et des fonctionnalités. Dans notre cas, l'API nous a permis l'extraction d'un ensemble d'offres emploi sans passer par des méthodes de web scrapping qui nous auraient demandé des prétraitements supplémentaires.

1.3 Description des données récupérées

Notre choix s'est porté sur les métiers liés au domaine du big data, nous avons donc décidé de récupérer 900 offres dont le mot clés est "data". Les données de l'API sont récupérées aux formats JSON, une fois celui-ci transformé en data frame, nous avons pu commencer l'étape de prétraitement des données.

2 Pré-traitement des données et alimentation de la base de données

2.1 Nettoyage des données

Le premier constat que nous avons pu faire en récupérant les offres d'emploi, c'est le nombre de données manquantes. En effet, le site pôle emploi met en place des champs que l'employeur doit définir (compétences,

qualification, niveau requis, etc...). Mais, nous nous sommes rendu compte que les employeurs renseignaient quasiment toutes les informations de leurs offres d'emploi dans la variable description, surement par facilité, afin de pouvoir effectuer des "copier-coller" de ces offres sur plusieurs sites de diffusion. Notre choix face à ce premier constat a été de supprimer les variables vides, celles qui présentaient très peu d'informations alors qu'elles étaient pertinentes pour l'analyse comme la qualification du poste, et celles qui nous semblaient peu pertinentes dans la réalisation de notre processus d'analyses.

Dans un second temps, nous nous sommes intéressés au pré-traitement des variables à forte composante textuelle. Il s'agit des variables "Description de l'offre", "Intitulé du poste" et "Description de l'entreprise". À travers une conjonction entre la recherche par "Expressions régulières" et la fonction de substitution "gsub", nous avons appliqué un certain nombre de modifications sur les variables citées plus haut :

- Suppression des sauts de ligne
- Suppression de toutes les ponctuations [[:punct :][:blank :][:space :]]
- Conversion du texte en minuscule
- Suppression de tous les mots de taille 1 et 2
- Suppression de tous les mots qui sont constitués de chiffres : "087", "12ème",...
- Remplacement de tous les longs espaces par 1 seul espace pour permettre une bonne tokénisation

Par la suite, nous avons recodé la variable "Salaire.libellé". Comme exemple pour cette variable, on a :

Annuel de 40000,00 Euros à 70000,00 Euros sur 12 mois

Mensuel de 1700,00 Euros à 1900,00 Euros sur 12 mois

On peut voir que sur certaines offres, l'employeur renseigne la plage de salaire annuelle qu'il met à disposition pour le job alors que d'autres renseignent la plage de salaire mensuelle. Afin d'uniformiser les informations de cette variable, nous avons créé une fonction qui s'est chargée de calculer le salaire mensuel moyen pour toutes les offres ayant la variable "salaire ;libellé" renseignée.

2.2 Organisation des données

Les variables que nous avons conservé suite au prétraitement sont :

- intitule : intitulé de l'offre
- description : description de l'offre
- latitude
- longitude
- salaire-moyen
- expérience : expérience exigée pour candidater sur l'offre
- ville
- département
- région
- entreprise
- typecontrat : CDD, CDI
- secteur
- date

2.3 Modèle conceptuel des données et alimentation des tables

L'ensemble de nos données sont stockées dans un SGBD libre sous la forme d'un entrepôt de données comprenant une table de fait, des dimensions et des niveaux de hiérarchies. Le modèle conceptuel des données hébergé sur PhpMyAdmin est visible ci-dessous.

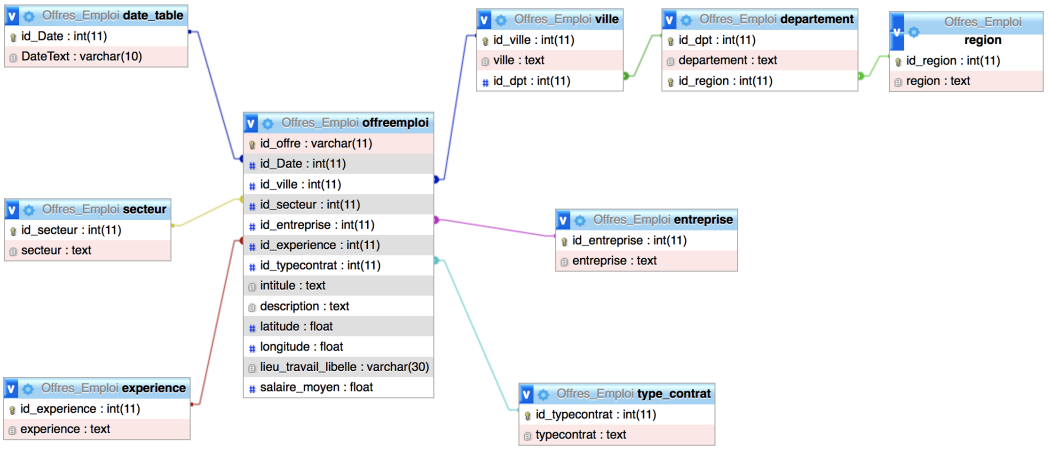

Figure 1 : Modèle conceptuel des données

La table de fait nommée "offre d'emploi" comprend toutes les clés primaires des différentes dimensions qui lui sont reliées. Une dimension représente un axe d'analyse que nous avons souhaité explorer. Ainsi, nous analysons les offres d'emploi selon l'entreprise, le secteur d'activité, le type de contrat proposé, le niveau d'expérience demandé, la date de parution ainsi que la localisation. Ce dernier axe se compose de niveaux hiérarchiques, car nous souhaitons réaliser une analyse régionale des offres d'emploi en s'appuyant sur les données à niveau départemental déjà renseignées sur le site de Pôle Emploi.

Les tables ont été remplies à l'aide de fichier .csv créés sur Rstudio. Chaque dataframe créé sur R nous a permis d'alimenter les bases de données, un fichier .csv correspondant à une table précise.

3 Analyses textuelles

3.1 Topic Modeling

Concernant le topic modeling, nous avons utilisé la méthode LDA, c'est une méthode non supervisée qui permet de décrire des collections de documents, ou d'autres types de données discrètes. Pour cela, LDA cherche à découvrir des structures thématiques cachées dans des vastes archives de documents.
LDA est un modèle Bayesien hiérarchique à 3 couches : chaque document est modélisé par un mélange de topics (thèmes) qui génère ensuite chaque mot du document. La structure des documents du corpus peut être déterminée par des techniques d'inférence.

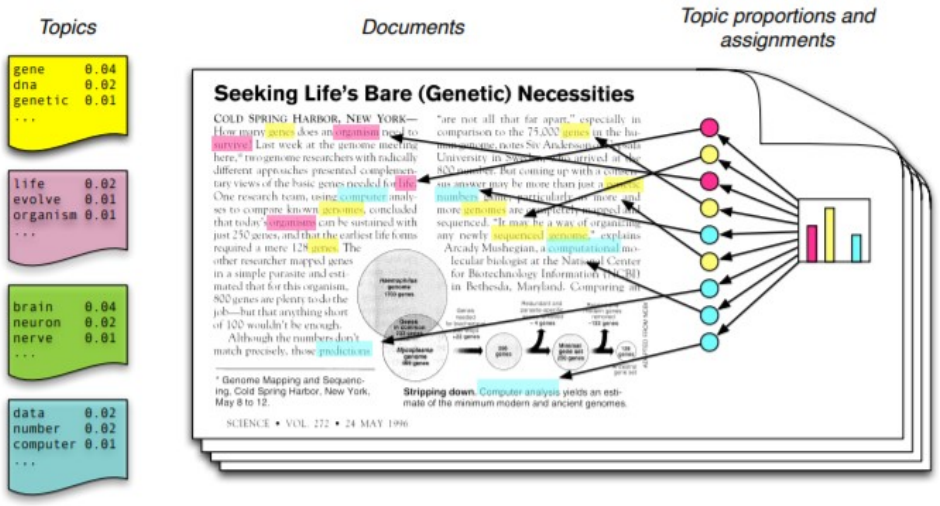

Figure 2 : LDA

Les topics renvoyés par le modèle LDA sont chacun constitués d'une liste de mots. Ces mots possèdent des probabilités d'appartenance. Plus un mot a une probabilité élevée dans un topic, plus cela veut dire qu'il est caractéristique de ce topic. La LDA se base sur 2 principaux calculs de probabilité pour créer des topics de plus en plus cohérents. Il s'agit de :

- La probabilité d'apparition d'un topic dans un document
- La probabilité d'apparition d'un mot dans un topic

3.2 Text categorization

Dans cette partie, nous avons utilisé le modèle statistique LSA (analyse sémantique latente) qui est un procédé de traitement des langues naturelles développé par les laboratoires Bellcore en 1989 pour de la recherche documentaires. LSA permet d'établir des relations entre un ensemble de documents et les termes qu'ils contiennent. Pour cela, le modèle prend en entrée la matrice termes documents, et par la suite il transforme la matrice des occurrences en une relation entre les termes et une relation entre les documents.

4 Application R Shiny

4.1 Présentation de l'application créée

L'application R Shiny s'articule directement sur la base de données que nous avons créé. L'utilisateur peut donc avoir accès à l'ensemble des données disponible sur le SGBD hébergé sur PhpMyAdmin et grâce aux requêtage, nous pouvons répondre à l'ensemble de ces besoins sur l'application R Shiny. Notre application permet de visualiser les offres d'emploi proposées par le site de Pôle Emploi durant les mois précédents et comportant le mot "data".

4.1.1 Premier Onglet - Recherche des offres

L'application s'articule selon quatre onglets : un premier onglet offre la possibilité à l'utilisateur d'appliquer un certain nombre de filtres sur les offres d'emploi proposées par le site de Pôle Emploi, afin d'obtenir

les résultats qui l'intéressent. La page d'accueil de l'application est visible ci-dessous.

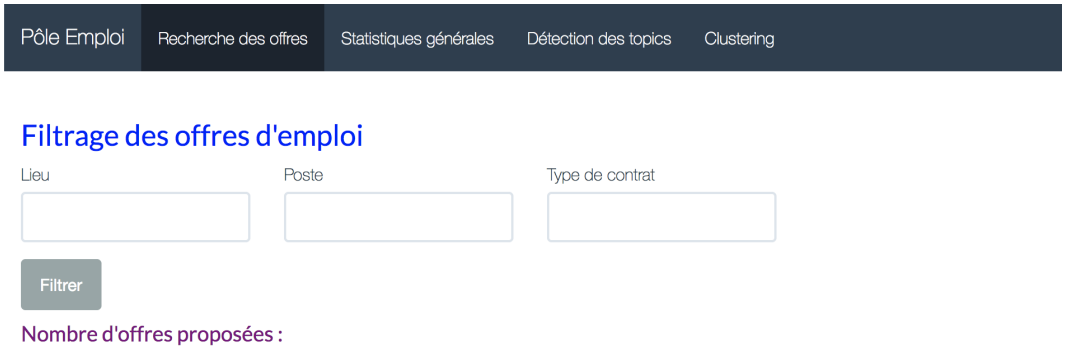

Figure 3 : Page d'acceuil de l'application R Shiny

Ces différents filtres offrent la possibilité de renseigner un lieu, un poste et un type de contrat préféré. Il est possible de filtrer les offres sur l'ensemble de ces critères de recherche ou seulement sur quelques-uns, ou tout simplement de choisir d'afficher toutes les offres sans application de filtres. Lorsque l'utilisateur appuie sur le bouton "Filtrer", les offres d'emploi correspondant à sa requête sont affichées ainsi que le nombre d'offres associé. L'utilisateur a alors accès à toutes les informations concernant les offres d'emploi comme l'intitulé du poste, l'entreprise et le lieu de travail ainsi qu'une description de l'offre indiquant ce qu'attend l'entreprise en termes de travail et ce qu'elle cherche en termes de profil. La figure ci-dessous est un exemple de requête que l'on peut rechercher.

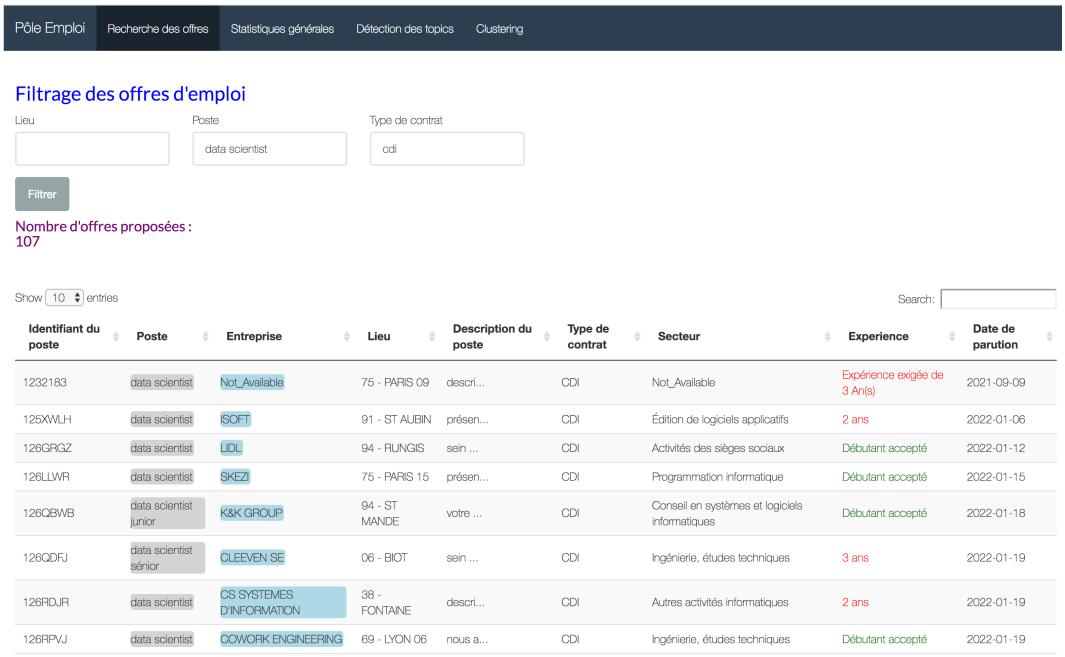

Figure 4 : Exemple de filtrage sur les offres d'emploi de Pôle Emploi

De plus, pour plus de lisibilité, l'utilisateur aperçoit seulement le début de la description du poste, mais lorsqu'il survole la cellule à l'aide de sa souris, il accède à l'intégralité de la description sous forme d'info-bulle et cela pour chacune des offres. Enfin, dans le cas où l'utilisateur serait un jeune débutant sur le marché du travail, un passage du niveau d'expérience en rouge a été rajouté lorsque l'entreprise demande au minimum deux ans d'expérience et dans le cas contraire le niveau d'expérience apparaît en vert. Quelques fonctionnalités sur l'affichage des offres permettent à l'utilisateur d'afficher un nombre d'offres souhaité par page (10, 25, 100 offres...) et également de réaliser une recherche par mot clés pour déterminer quels sont les offres qui pourraient contenir un mot en particulier. On peut prendre l'exemple du mot clé "spark" et le taper dans la barre de rechercher associée. On peut voir qu'il est contenu dans plusieurs offres de Pôle Emploi. Il est important de noter que les offres d'emploi filtrées par l'utilisateur sur ce premier onglet servent dans la suite de l'application. Ces mêmes offres seront analysées et traitées sur les onglets suivants.

4.1.2 Deuxième Onglet - Statistiques générales

Un deuxième onglet affiche diverses analyses statistiques générales sur la fréquence des mots les plus rencontrés dans les offres d'emploi ainsi qu'une carte présentant la répartition des offres proposées en France.

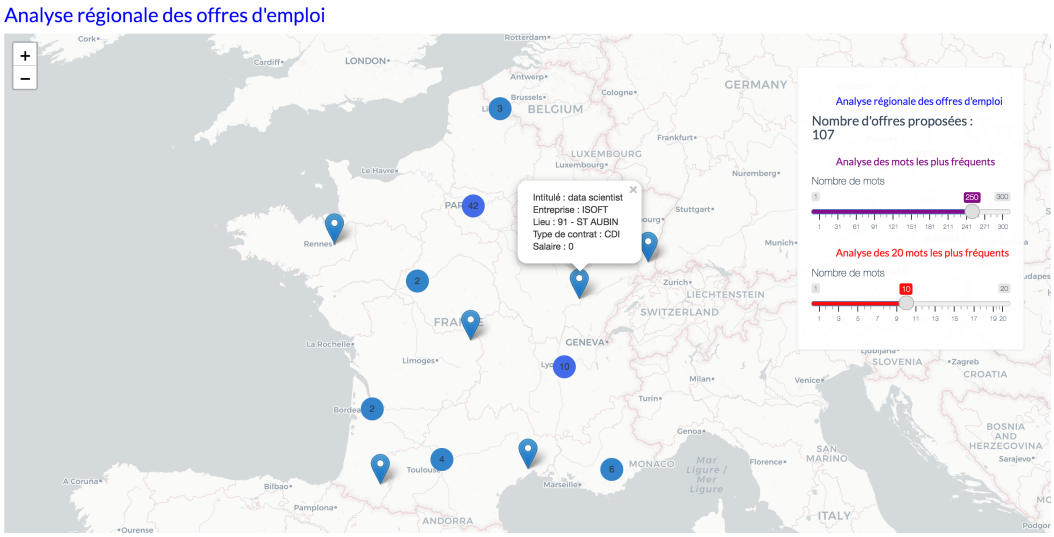

Figure 5 : Visualisation des offres d'emploi par région de France correspondant aux offres filtrées de l'onglet précédent

Sur cette carte de France, on peut visualiser le nombre d'offres d'emplois par région en France, comportant toujours le mot "data". Lorsque l'on souhaite apercevoir les offres d'emploi d'une région en particulier, il suffit de zoomer sur la région en question. Les curseurs modélisant chacun une offre d'emploi vont se démultiplier à mesure que l'utilisateur va zoomer sur la carte de France. Des informations sur l'offre d'emploi vont apparaître on clique sur le curseur associé. On a alors accès à l'intitulé de l'offre, l'entreprise, le lieu de travail, le type de contrat ou encore le salaire lorsque celui a été renseigné. En dessous de cette carte, deux graphiques sont disponibles avec pour chacun un filtrage sur les données accessible depuis la partie droite de l'application. Ce panel de réglage peut être déplacé sur la page internet en le sélectionnant, s'il vient à cacher certaines informations sur les graphiques. Cependant, ce panel devient transparent lorsque la souris de l'utilisateur ne le survole pas, autrement dit lorsque l'utilisateur n'a plus besoin d'effectuer de réglage sur ce panel.

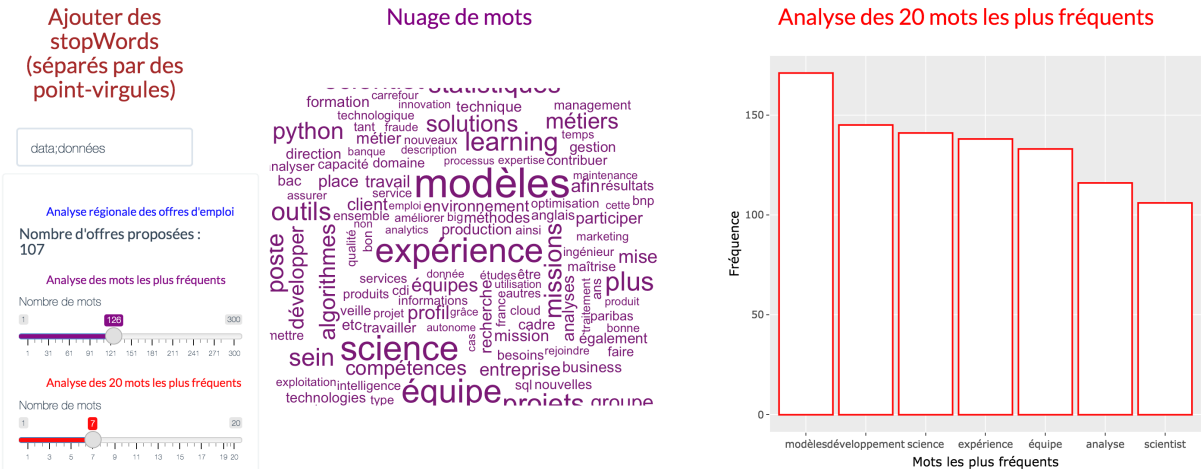

Figure 6 : Analyse des mots les plus fréquents dans les offres d'emploi filtrées

Concernant les statistiques générales sur les offres d'emploi, il s'agit dans un premier temps d'un nuage de mots présentant les mots les plus fréquents parmi l'ensemble des offres sélectionnées par l'utilisateur dans l'onglet précédent. Un curseur permet à l'utilisateur de choisir le nombre de mots qu'il souhaite afficher sur le nuage de mot. Un histogramme permet d'afficher la fréquence des mots parmi ceux revenant le plus souvent, il s'agit là du top 7 des mots les plus fréquents. Là aussi, l'utilisateur peut choisir le nombre de mots qu'il souhaite afficher entre 1 et 20 mots. Enfin, l'application offre la possibilité d'ajouter des stopwords afin que ceux-ci ne figurent pas dans les mots les plus fréquents sur le nuage de mots et l'histogramme des fréquences de mots. En effet, les mots comme 'données ou 'data' sont des mots qui reviennent particulièrement souvent et qui faussent l'analyse sur les mots les plus fréquents dans les offres d'emploi.

4.1.3 Troisième Onglet - Détection des topics

Le troisième onglet propose à l'utilisateur d'implémenter la méthode de Latent Dirichlet Aloccation (LDA) sur les données de description des offres d'emploi. Dans un premier temps, l'application propose à l'utilisateur de déterminer la meilleure métrique ainsi que le meilleur nombre de topics à instancier sur la LDA.

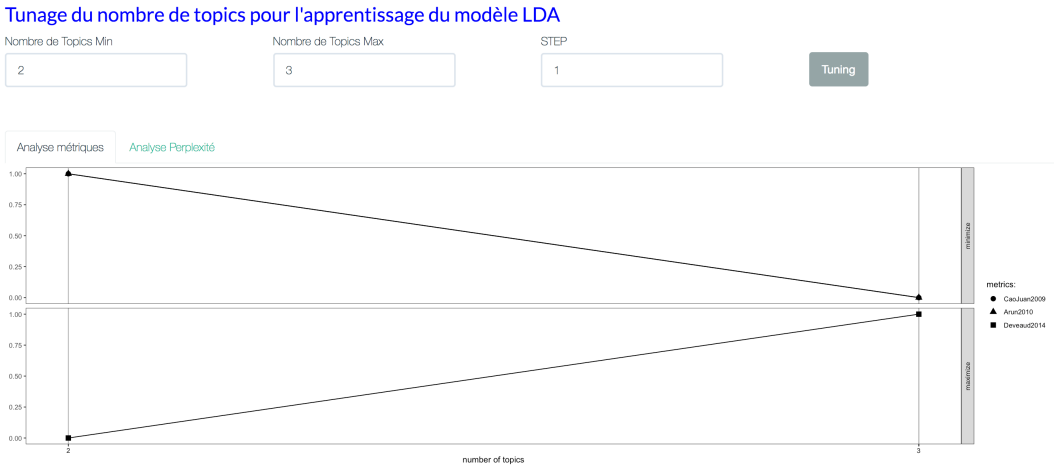

Figure 7 : Tunage du nombre de topic pour l'apprentissage du modèle LDA - Analyse des métriques

Sur la figure ci-dessus, on permet à l'utilisateur de choisir le nombre de topic minimum et maximum que l'on souhaite instancier dans différents apprentissages de modèle LDA. Le but est d'observer sur le premier cadran "Analyse métriques", la métrique qui conduit à l'obtention de meilleurs résultats. Différentes métriques sont utilisables dans un modèle LDA. On a par exemple "CaoJuan2009", "Arun2010", "Deveaud2014", etc. Sur cadran supérieur avec le label "minimize", il s'agit de repérer la métrique qui présente une forte décroissance au fur et à mesure que le nombre de topics augmente. Sur cadran inférieur avec le label "maximize", il s'agit de repérer la métrique qui présente une forte croissance au fur et à mesure que le nombre de topics augmente. Le but final est de déterminer la métrique optimum à utiliser dans l'apprentissage du modèle LDA.

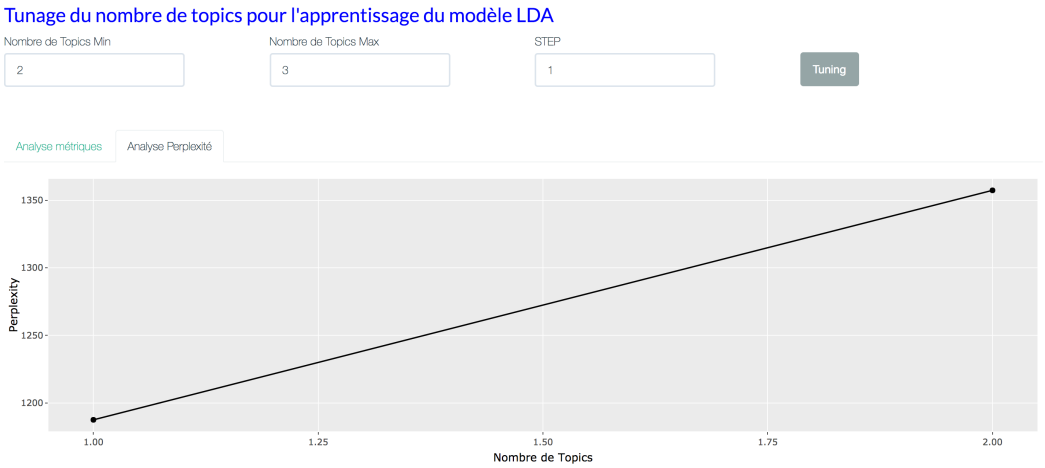

Figure 8 : Tunage du nombre de topic pour l'apprentissage du modèle LDA - Analyse perplexité

Le cadran "Analyse perplexité" est censé permettre à l'utilisateur de choisir le meilleur nombre de topics pour le modèle LDA. PAr définition, plus la perplexité a une faible valeur, plus le modèle est performant. C'est-à-dire qu'il a des topics de plus en plus cohérents. On permet donc à l'utilisateur à travers ce graphique de choisir le nombre de topics pour lequel il a la plus faible valeur de perplexité.

En se basant sur le nombre de topics et la métrique déduite de l'analyse précédente, l'utilisateur peut désormais lancer l'apprentissage d'un modèle LDA sur les descriptions d'offres chargées sur le premier onglet et visualiser le contenu des topics obtenus.

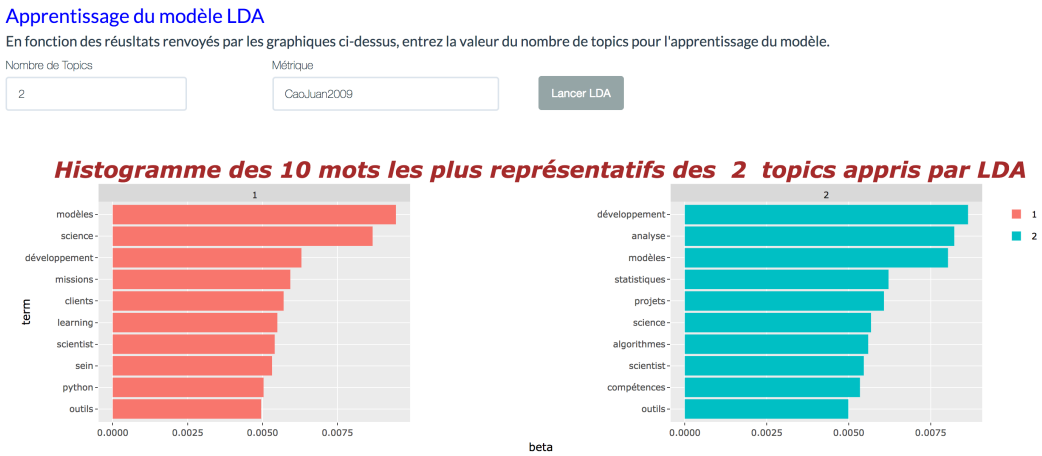

Figure 9 : Apprentissage du modèle LDA

Il s'agit là d'une représentation sous forme d'histogramme des 10 mots les plus représentatifs des topics appris par le modèle LDA. On obtient donc autant de groupes d'histogrammes qu'il n'y a de topics renseignés par l'utilisateur. Dans le meilleur des scénariios, les 10 meilleurs mots présents dans chaque topic doivent permettre de déduire un thème pour ce "topic". Par exemple "les métiers du cloud", "l'intélligence artificielle", etc.

4.1.4 Quatrième Onglet - Clustering

Le quatrième onglet permet à l'utilisateur de réaliser un clustering sur les mots provenant des descriptions des offres et sur les offres en eux-même. Ayant une matrice document-Term contenant plus de 13000 mots, nous avons décidé de réduire la quantité de mots utilisés pour le clustering avec LSA. En l'occurrence, nous avons décidé de nous baser sur les 10 mots les plus pertinents de chacun des topics renvoyés par une LDA. Ceci nous permettait de garantir un clustering sur des mots ayant beaucoup de sens dans la description des offres. C'est la raison pour laquelle sur cet onglet, on demande à l'utilsateur de lancer une LDA sur un nombre de topics au choix, puis le modèle LSA se basera sur les résultats renvoyés par LDA pour réaliser le clustering.

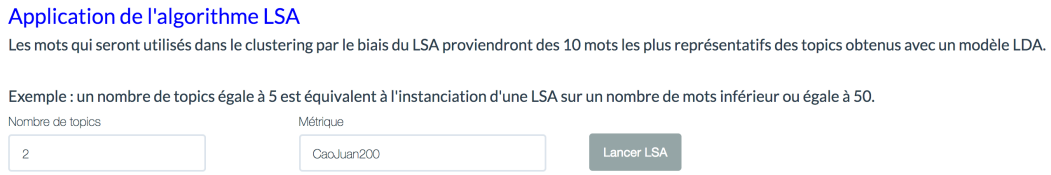

Figure 10 : Application des paramètres de la méthode LSA

Lorsque l'utilisateur appuie sur le bouton "Lancer LSA", on obtient une projection des mots d'une part et des identifiants des offres d'autre part sur les deux premiers axes. Le but est alors de repérer les mots ou les identifiants d'offres qui sont projetés sur les mêmes espaces en vue de déduire des thèmes. Les identifiants d'offres constituent des informations très vagues. Il est alors conseillé à l'utilisateur de se servir de la barre de recherche sur le tableau du premier onglet afin d'analyser les offres qui à priori se retrouvent projetées sur le même espace.

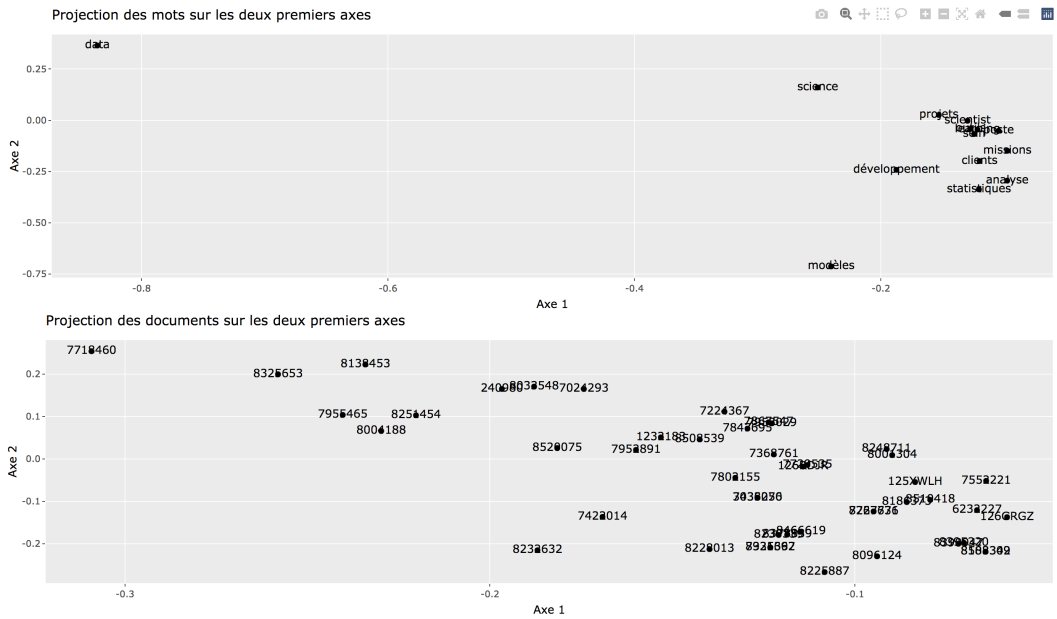

Figure 11 : Projection des mots et des identifiants des offres sur les deux premiers axes

4.2 Architecture du code de l'application

L'architecture du code se sépare en deux principales fonctions : une première permettant de créer une mise en page de l'application avec le principe html. Il s'agit de la partie de l'interface qui sera visible pour l'utilisateur. Elle est appelée 'ui' pour 'User Interface'. De même, elle permet de récupérer des informations choisies par l'utilisateur par le biais de 'textInput' ou 'sliderInput'. Il s'agit des fonctions qui vont récupérer respectivement une chaîne de caractères entrée par l'utilisateur ou sélectionner une valeur numérique dans une plage à partir d'un widget de type curseur. Dans la fonction ui, nous avons intégré une barre de navigation à l'aide de la fonction navbarPage() afin de permettre à l'utilisateur de naviguer facilement d'un onglet à l'autre ; chaque onglet étant défini avec la fonction tabPanel(). Les graphiques ou cartes sont affichées à l'aide de fonction output correspondant à des éléments de sortie de tracé dans la fonction server de notre application, deuxième fonction principale. De plus, l'affichage de certains graphiques apparaît uniquement lorsque l'utilisateur clique sur le bouton associé. Il s'agit là d'un bouton d'action dont la valeur est initialement de zéro, et s'incrémente de un à chaque fois qu'il est pressé. Ainsi, tant que l'utilisateur ne clique pas sur ce bouton, les graphiques ne vont pas s'afficher. Cette condition est elle défini dans la seconde fonction du code source de notre application R Shiny.
En effet, cette seconde fonction correspond à l'arrière-plan de l'interface R Shiny, c'est-à-dire toutes les fonctions qui vont servir au filtrage des données renseignées par l'utilisateur puis à leur modélisation sous forme de graphique. Il s'agit donc là de la partie de l'interface dont l'utilisateur n'a pas accès. Dans un premier temps dans cette fonction, on va d'abord connecter l'interface à la base de données créée, uniquement lorsque l'utilisateur chargera les offres d'emploi en cliquant sur le bouton 'Filtrer'. Puis, on réalise ce filtrage à l'aide des informations qu'il aura tapées dans le textInput() évoqué précédemment et en utilisant le langage SQL pour effectuer des requêtes directement sur la base de données. Ensuite, chacune des fonctions output sont créées afin de modéliser les différents algorithmes et graphiques de notre application. Par exemple, pour l'affichage de la table de données présentant les différentes offres d'emploi, on réalise une version réactive de la fonction donnée qui renvoie un cadre de données (ou une matrice), qui sera rendu avec la bibliothèque DataTables.

Enfin, le démarrage de l'application nécessite la ligne de code suivante :

```
shinyApp(ui=ui, server=server)
```

La fonction shinyApp() fait alors appel aux deux principales fonction décrites ci-dessus.

4.3 Analyse du corpus : Etude de cas

Afin de proposer une analyse de notre corpus se composant des offres d'emploi du site de Pôle Emploi, nous allons étudier un cas en particulier. Dans le premier onglet, nous allons renseigner les filtres suivant : 'data' pour le nom du poste et 'cdi' pour le type de contrat comme montré ci-dessous.

Figure 12 : Filtrage sur les offres d'emploi pour l'étude de cas

Nous obtenons alors 718 offres parmi les 889 de notre base de données.

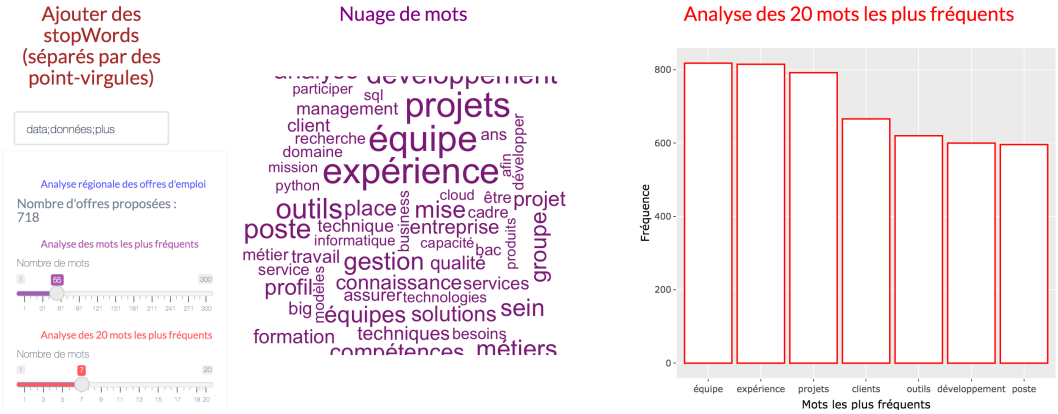

Figure 13 : Statistiques générales pour l'étude de cas

Sur le deuxième onglet, on identifie des stopwords à ajouter à notre liste de stopwords initialement implémentée dans notre code source. Nous y ajoutons donc les mots 'data', 'données' et 'plus' car ils ne nous serviront pas dans la recherche de cluster de mot dans l'analyse des offres d'emploi.

À l'aide de l'apprentissage de la méthode LDA avec comme nombre de topics 5, nous obtenons les résultats suivants.

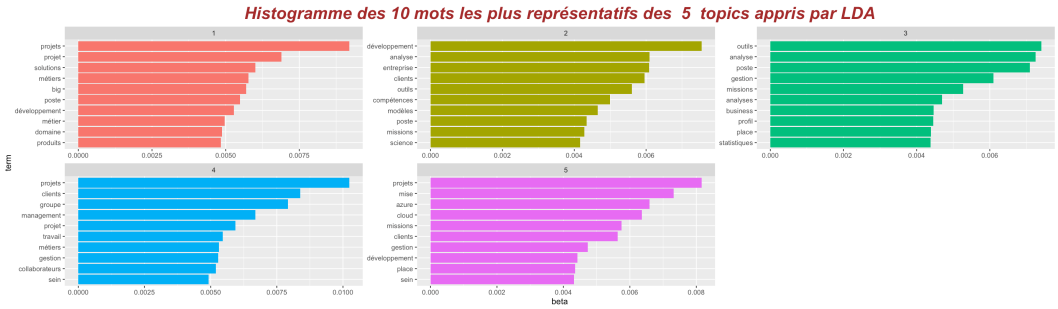

Figure 14 : Apprentissage de la méthode LDA pour l'étude de cas

Par exemple, nous pouvons voir que le topic numéro 3 semble regrouper les métiers de la statistique, car les mots les plus fréquents caractérisant ce topic sont les mots 'statistique', 'analyse', 'outils'. A contrario, le topic numéro 5 lui semble être caractérisé par des mots tournés sur le thème du cloud comme 'azure', 'cloud'. Afin d'étayer ces propos, nous réalisons une LSA sur l'onglet suivant de notre application.

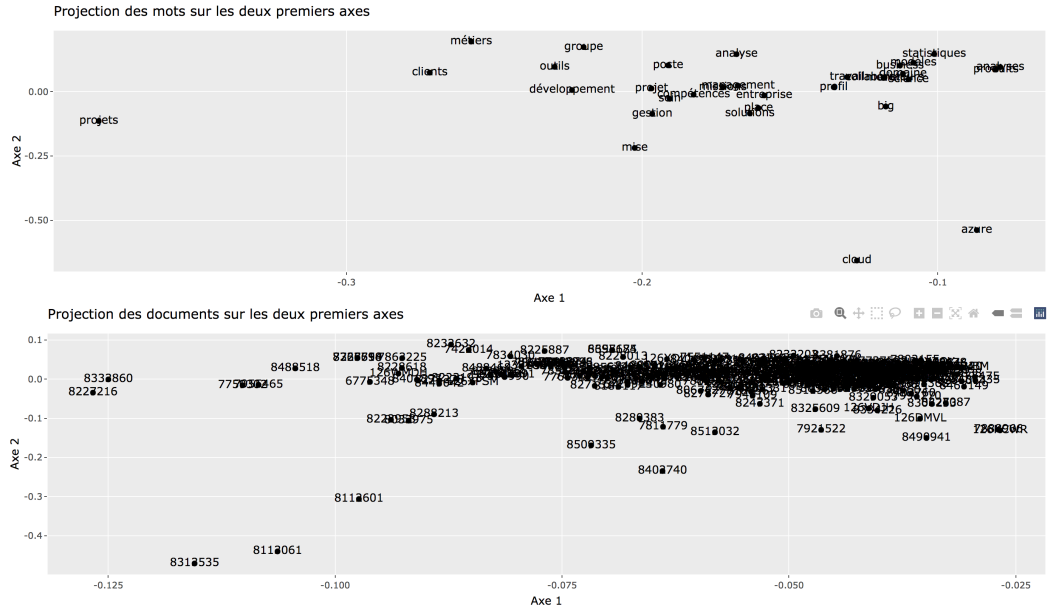

Figure 15 : Apprentissage de la méthode LSA pour l'étude de cas

Sur la projection des mots sur les axes, nous pouvons constater à nouveau que les mots 'azure' et 'cloud', caractéristique du topic 5, semble être particulièrement prochent sur la projection de l'axe 1 et de l'axe 2. De même que les mots 'statistique', 'business', 'analyse', 'modèle' sont projetés assez proche sur l'axe 1 et l'axe 2.

Sur la seconde projection correspondant aux documents, on peut voir que deux offres d'emploi semblent se démarquer des autres. Vérifions cela en revenant sur le premier onglet pour déterminer quelles sont leurs similitudes.

Figure 15 : Vérification du poste des deux offres étudiées en particulier

D'après notre vérification dans les offres d'emploi disponible sur le premier onglet, nous découvrons qu'il s'agit de deux offres ayant comme thème principal le cloud. De plus, les mots 'azure' et 'cloud' semblent se projeter sur les axes de la même manière que les deux offres d'emploi étudiées. Nous pouvons donc dire que les mots 'cloud' et 'azure' sont caractéristiques des offres d'emploi 8113061 et 8313535.

5 Difficultés rencontrées et perspectives

5.1 Difficultés rencontrées

Afin d'arriver au résultat de cette application R Shiny, nous avons dû faire face à de nombreux problèmes.

- L'un des premiers problèmes a été celui d'extraire les offres d'emploi du site de Pôle Emploi de manière structurée afin de disposer de données plus facilement modulable. En effet, nous avons pu constater que certains sites comme "Indeed" présentait plusieurs restrictions quant à l'utilisation de leur API pour l'extraction de données de leurs bases. En l'occurrence, l'impossibilité d'effectuer plusieurs requêtes d'extraction au même moment. Le site sur lequel on s'est finalement basé, "Pôle emploi", présentait une limite d'extraction de 150 offres par requête. Nous avons dans un premier temps tenté une extraction simultanée de plus de 150 offres ce qui s'est résolu par un échec. Nous nous sommes alors résolu à lancer séparément plusieurs requêtes d'extraction. Puis, nous avons fait un "merge" des données récupérées.
- Nous avons voulu intégrer une technique de Lemmatisation au sein de notre application. C'est une technique de text mining qui permet de réduire les mots d'un corpus à leur plus petite racine ayant du sens. Par exemple les mots ("run", "ran", "running") seraient transformés en "run", leur lemme commun. Cependant, le chargement du dictionnaire de mot en français prend énormément de temps. Nous avons alors jugé qu'il était inapproprié de l'intégrer dans notre application.
- Un problème exceptionnel s'est produit avec la librairie "stopwords". Nous nous sommes servi de cette librairie pour télecharger les stopwords en français afin de filtrer les données textuelles et ne garder que les mots pertinents. Au départ, cette librairie fonctionnait parfaitement bien, mais quelque jours avant le rendu de ce projet, la récupération des stopwords ne marchait subitement plus. Après quelques recherches sur les forums, quelques indications nous ont conduits vers une ré-installation de la librairie qui visiblement avait subi des modifications et donc ne fonctionnait plus.

5.2 Perspectives

Plusieurs améliorations sont possibles sur cette application RShiny.

- Dans un premier temps, on pourrait penser à la gestion des erreurs. Nous n'avons pas géré les situations où l'utilisateur entrerait des valeurs erronées dans les champs de notre application. En cas d'erreur, ce dernier se retrouve obligé de relancer l'application et de recharger le jeu de données.
- Ensuite, sur le premier onglet, dans les champs dédiés aux filtres, la possibilité d'entrer juste quelques caractères et qu'en fonction, on propose une liste de choix pertinents de filtrage est une bonne piste pour rendre l'application plus user-friendly.

- Sur l'analyse du corpus, vous pourrez constater que l'on se retrouve avec un mot et son pluriel, ce qui en réalité renvoie à un même sens. Une lemmatisation automatique appliquée sur les variables à forte composante textuelle serait une option très intéressante pour l'analyse des corpus.

Conclusion

Pour finir, ce projet nous a permis d'approfondir nos compétences en programmation sur R, nos connaissances en matière de création d'une application avec R Shiny et notre faculté à analyser un corpus à l'aide de technique de text mining. Il nous aura permis de développer de réelles compétences en matière de gestion de projet (répartition de tâches, etc.) et donc d'appréhender la réalisation de projet au sein d'une équipe. Compétences qui seront réellement utiles dans le monde professionnel.