C

Projet de fin d'étude

Airbus Helicopters

Développement d'un moteur de recherche

2019-2020

Tuteur pédagogique: Pierre MICHEL

Tuteur professionnel: Flavien RICHE

Etudiants: Arthur Bartoli

Djiby Balde

Yannis El Abounni

Théa Gilles

Tatiana Maia

Inès Ouaret

Sommaire

1. Introduction

Actuellement en troisième et dernière année de Magistère Ingénieur Economiste, nous sommes répartis en plusieurs groupes composés de 6 personnes pour travailler sur un Projet de Fin d'Etudes (PFE). Le but de ce projet est de nous confronter à la réalité terrain au sein d'une entreprise. Notre groupe a été sélectionné pour travailler avec Airbus Helicopters basé à Marignane (13).

1.1. Qui est Airbus Helicopters?

Airbus Helicopters, d'ancien nom "Eurocopter", a été créé le 1er Janvier 1992 grâce à la fusion des divisions hélicoptères des sociétés allemande «Deutsche Aerospace » et française « Aerospatiale ».

Avec un chiffre d'affaires de 5.9 Milliards d'euros et près de 20 000 employés, la branche Airbus Helicopters est aujourd'hui le premier fabricant d'hélicoptères sur le marché civil, militaire et parapublic. En effet, ce chiffre d'affaires est partagé entre le service militaire, avec une part de 46% et le marché civil qui représente 54% du chiffre d'affaires du groupe.

Les pays fondateurs, à deux, englobent 70% de l'effectif total. Le siège localisé donc à Marignane (France) compte avec environ 8 000 effectifs et le site de Donauwörth (Allemagne) compte avec 5 500 effectifs.

Airbus Helicopters est présent dans 148 pays par l'intermédiaire de ses 30 Custumers centres et sites affiliés. Chaque site, indépendamment de la localisation géographique, a pour but de répondre rapide et efficacement aux besoins de ses clients.

Airbus Helicopters développe, produit, commercialise et assure la maintenance d'hélicoptères adaptés aux différents besoins des clients. Ainsi, l'entreprise couvre une palette d'activités allant du développement (recherche, conception), au service client (commercialisation, assistance technique, révision, réparation).

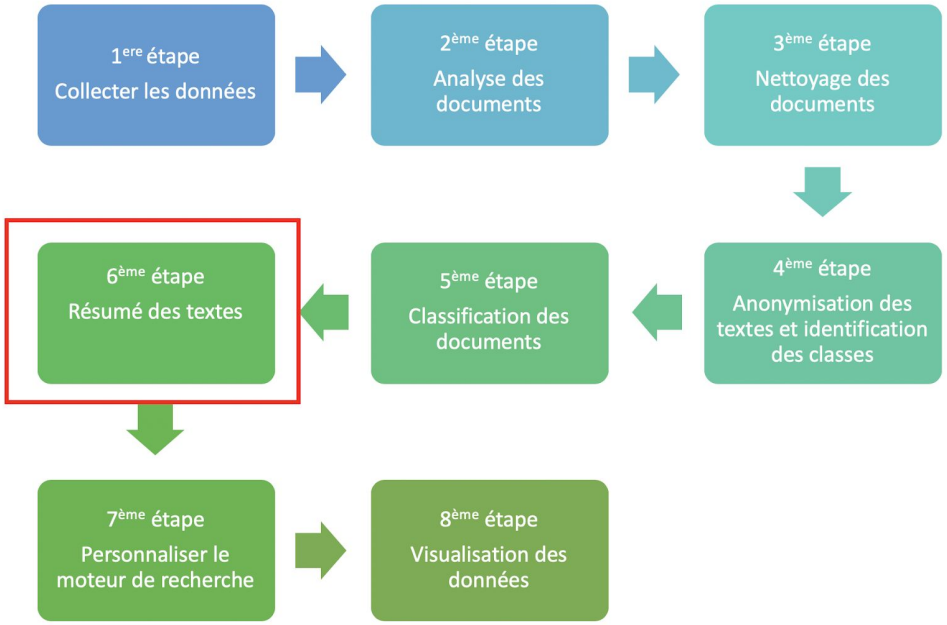
Parmi les 5 continents, Airbus Helicopters compte avec 19 centres de formation, 4 hubs de support technique, 8 hubs logistiques – stocks régionaux et centraux, et environ 100 centres de Services (31 centres en Europe).

Grâce à ces appareils, Airbus Helicopters posséde une forte pénétration des marchés publics et parapublics grâce à une gamme d'hélicoptères civils très diversifiée.

1.2. Quelle est notre problématique ?

Pour ce projet, nous travaillons en collaboration avec le service de la transformation digitale d'Airbus Helicopters. Les principaux objectifs de ce service sont d'accélérer la numérisation de l'entreprise, créer de la valeur avec des produits innovants, centraliser les données et avoir un rôle transversal dans la numérisation d'Airbus. Notre groupe va intervenir sur un projet qu'ils ont commencé et qui va permettre de construire un moteur de recherche de documents.
Voici donc les étapes à suivre :

Nous pouvons voir sur ce schéma que notre travail débute à l'étape 6. Les 5 étapes précédentes ont été faites par les équipes du service de la transformation digitale. Ils ont d'abord commencé par récupérer des documents parlant d'Airbus. Ils ont ensuite récupéré seulement le texte (ils ont supprimé les photos par exemple). Et enfin, le texte a été nettoyé pour qu'il n'en reste que le texte brut.
Notre groupe intervient pour résumer les documents (6ème étape). Le but est de prendre un ensemble de documents et de récupérer seulement les phrases les plus pertinentes qui résument au mieux les documents afin de faciliter le travail des ingénieurs ou autres.
Une fois cette étape terminée, les équipes d'Airbus Helicopter pourront continuer le processus.
Airbus Helicopters nous a fourni 30 documents, rédigés en anglais, répartis en 3 thèmes (10 documents par thème). Ces documents parlent de trois hélicoptères différents. Nous pouvons donc commencer nos recherches pour trouver une solution pour résumer les documents automatiquement.



1.3. Recherches préliminaires

1.3.1. Natural language preprocessing : text summarization

En commençant nos recherches sur “text summarization”, nous nous sommes aperçus que c’était lié avec le “Natural Language Processing” (NLP). Il est important de comprendre ce qu’est le NLP pour ensuite pouvoir résumer des textes. Le NLP est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle, qui vise à créer des outils de traitement de la langue naturelle pour diverses applications. Le traitement du langage naturel comprend de nombreuses techniques différentes pour interpréter le langage humain, allant des méthodes statistiques et d'apprentissage automatique aux approches basées sur des règles et algorithmiques. Nous avons besoin d'un large éventail d'approches, car les données textuelles et vocales varient considérablement, tout comme les applications pratiques.

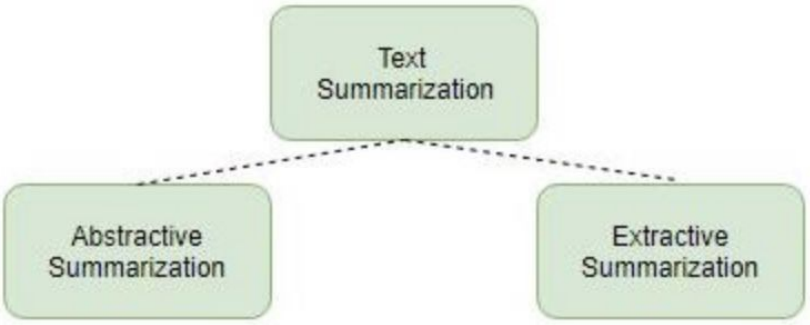
Les tâches de base du NLP incluent la tokenisation et l'analyse, la lemmatisation / stemming, l'étiquetage d'une partie du discours, la détection de la langue et l'identification des relations sémantiques. En termes généraux, les tâches de NLP décomposent le langage en morceaux élémentaires plus courts, essayent de comprendre les relations entre les morceaux et explorent comment les morceaux fonctionnent ensemble pour créer du sens. L'objectif primordial est de prendre la saisie du langage brut et d'utiliser la linguistique et les algorithmes pour transformer ou enrichir le texte de telle sorte qu'il offre une plus grande valeur.

1.3.2. Nos recherches

Nous avons débuté nos recherches et nous nous sommes aperçus, qu’il était nécessaire d’établir une première phase de nettoyage de données, comme pour tout projet de data science.
Nous avons donc crée plusieurs fonctions permettant de rendre les textes d’entrées compréhensibles par la machine.
Nous détaillerons mieux cette phase dans une prochaine partie.
Nous avons ensuite continuer nos recherches, et cela nous permis de voir qu’il existait dans ce domaine deux principales méthodes concernant la synthèse de document.

1.4. La théorie, qu’avons-nous appris des différents modèles et outils

En matière de résumé de texte il existe deux méthodes de résumé:



1.4.1. La méthode abstractive

La méthode abstractive sélectionne les mots sur la base de la compréhension sémantique même quand les mots n'apparaissent pas dans le texte d'origine.
A l'aide de techniques avancées de compréhension du langage naturel elle interprète et examine le texte dans le but de générer un nouveau texte plus court qui retranscrit les informations principales du texte d'origine.
Cette méthode peut être comparée à la façon dont les humains lisent un article de texte et le résument ensuite dans leur propre mot. En effet, elle tente de résumer les articles en sélectionnant un sous-ensemble de mots qui retiennent les points les plus importants.

Cette approche pondère la partie importante des phrases et l'utilise pour former le résumé. Différents algorithmes et techniques sont utilisés pour définir le poids des phrases et les classer en fonction de leur importance et de leur similarité.

- GPT-2

Durant nos recherches nous avons trouvé une méthode abstractive très performante. Il s'agit de GPT2.

GPT-2, est la deuxième version de GPT, Generative Pre-training Transformer. L'algorithme a été présenté en 2019 par OpenAI dans le papier *"Language Models are Unsupervised Multitask Learners".*

GPT-2 est donc un modèle de génération de texte, basé sur l'architecture Transformer, entrainé sur 40GB de textes récoltés sur internet. Le but de l'algorithme est de prédire le prochain mot de la phrase en tenant en compte les

mots précédents. Il peut être utilisé pour la traduction, la summarisation, la création de texte et il peut également répondre à des questions posées par l'utilisateur.

Due aux risques d'applications malveillantes de la technologie, OpenAI n'a pas mis le modèle à libre disposition des utilisateurs. Cependant ils ont mis le modèle dans une interface, *Talk Transformer,* à travers laquelle nous pouvons tester et utiliser le modèle.

Le modèle est donc basé sur un Transformer composé de 1,5 milliards de paramètres et entraîné sur des données de 8 millions de pages internet ayant donc des données diversifiées. Les créateurs du modèles affirment que GPT-2, en se basant sur un texte d'entrée, va être capable de générer des textes assez longs qui se ressemblent à des textes écrits par une personne.

GPT-2 est un modèle seq2seq donc il peut réaliser des résumés de texte. Dans ce qu'il concerne la summarisation, le modèle a été testé sur les articles de CNN et Daily Mail.

Voici un exemple de summarization de texte présenté dans le papier *"Language Models are Unsupervised Multitask Learners" :*

Article: *"Article:Prehistoric man sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave in modern day France 36,000 years ago.Now, with the help of cutting-edge technology, those works of art in the Chauvet-Pont-d'Arc Cave have been reproduced to create the biggest replica cave in the world.The man made cavern named the Caverne du Pont-d'Arc has been built a few miles from the original site in Vallon-Pont-D'arc in Southern France and contains 1,000 painstakingly-reproduced drawings as well as around 450 bones and other features...Cavemen and women sketched an incredible array of prehistoric beasts on the rough limestone walls of a cave 36,000 years ago and now a replica has been created (pictured)..."*

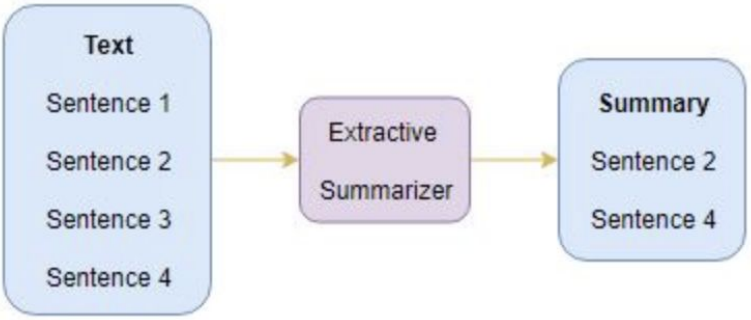
GPT-2 summary: *" The original site in Vallon-Pont-D'arc in Southern France is a Unesco World Heritage site and is the oldest known and the best preserved cave decorated byman. The replica cave was built a few miles from the original site in Vallon-Pont-D'Arc in Southern France. The cave contains images of 14 different species of animals including woolly rhinoceros, mammoths, and big cats."*

Les résumés purement extractifs donnent souvent de meilleurs résultats que les résumés automatiques. Cela s'explique par le fait que les méthodes de résumé abstractif font face à des problèmes tels que la représentation sémantique,

l'inférence et la génération du langage naturel, ce qui est relativement plus difficile que les approches fondées sur les données telles que l'extraction de phrases.

1.4.2. La méthode extractive

La technique de synthèse de texte extractif consiste à extraire des phrases clés du document source et à les combiner pour créer un résumé. L'extraction se fait selon la métrique définie sans apporter de modifications aux textes.



Les différentes approches de la synthèse extractive:

Approche positionnelle

Cette approche a été historiquement la première. Elle découle de l'observation que pour les articles scientifiques, la plupart des phrases informatives ont tendance à se trouver soit au début soit à la fin du document. Certaines de ces méthodes sont également complétées par des scores pour les mots qui sont calculés à partir de leur fréquence.

Résumé basé sur des graphiques

L'approche la plus populaire pour la synthèse. Elle consiste à créer un graphique sur les unités des documents (la plupart des méthodes utilisent des phrases comme unités de base) et ensuite à sélectionner des nœuds avec le PageRank (ou TextRank). L'algorithme du PageRank calcule la "centralité" des nœuds dans le graphe, ce qui s'avère utile pour mesurer le contenu informatif relatif des phrases. Le graphe tend à être construit en utilisant les caractéristiques des sacs de mots de phrases (généralement tf-idf).

Synthèse basée sur les centroïdes

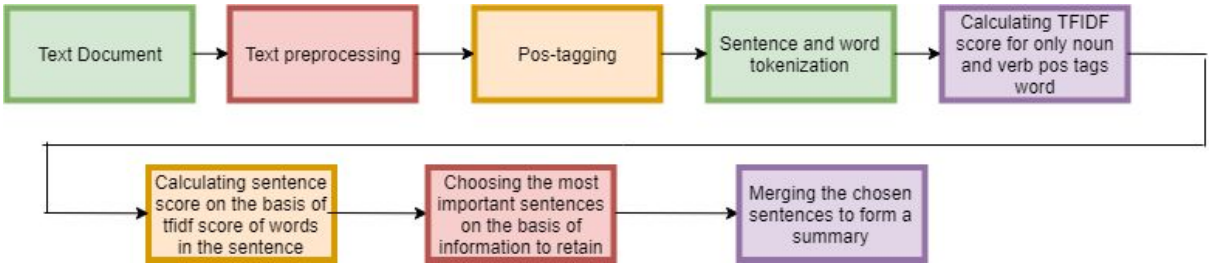
Supposons que nous ayons un moyen de représenter des phrases et des documents dans un espace de fonctionnalité. Le centroïde est défini comme le vecteur du document entier. Les phrases de synthèse sont sélectionnées en prenant des phrases qui ont des vecteurs similaires au vecteur centroïde.
A l'origine, cette approche a été suggérée pour le modèle Bag of Words (bien qu'elle utilise un certain prétraitement sur centroïde), mais récemment elle a été relancée dans dans le papier “Centroid Text Summarization through Compositionality of Word Embeddings”.

Résumé basé sur la méthode de la Distance Cosine

C’est une approche d'apprentissage non supervisée pour trouver les similitudes entre les phrases et les classer. L'un des avantages de cette méthode est qu'il n'est pas nécessaire d’entraîner et de construire un modèle avant de commencer à l'utiliser.
La similarité cosinus est une mesure de la similarité entre deux vecteurs non nuls d'un espace de produit intérieur qui mesure le cosinus de l'angle entre eux. Puisque nous représenterons nos phrases comme le bouquet de vecteurs, nous pouvons l'utiliser pour trouver la similarité entre les phrases.
Étapes de la méthode :
-Découpage du texte en phrase
-Construction de la matrice de similarité incluant chaque phrase du texte
-Classer les phrases dans la matrice de similarité
-Retenir les phrases qui sont les plus proches de toutes les autres phrases en termes de similarité cosinus
-Produire comme résumé les phrases retenues précédemment dans l’ordre de similarité.

1.4.3. Le TF-IDF

Cette méthode extractive extrait les phrases les plus importantes et les plus significatives du document texte en basant sur un certain nombre de procédures. La procédure de ce travail peut être matérialisée de la façon suivante.



Il s'agira donc, pour un document (ou topic dans notre cas) donné, de trouver les phrase les plus importants dans le document.

> document ⇒ trouver les mots les plus importants dans le documents ⇒ trouver les scores de phrase sur la base de l'importance des mots ⇒ choisir les phrases les plus importantes correspondantes

Term Frequency–Inverse Document Frequency (TF-IDF) est une mesure numérique qui sert à noter l'importance d'un mot dans un document en fonction de la fréquence à laquelle ce mot apparaît dans ce document et dans un ensemble donné de documents. Ainsi, il s'agira donc de donner un score aux mots en fonction de leurs fréquences d'apparition dans le document.

$$TF(w) = \frac{nombre\ de\ fois\ que\ w\ apparaît\ dans\ un\ document}{nombre\ total\ de\ terms\ dans\ le\ document}$$

$$IDF(w) = log_{10}(\frac{nombre\ total\ de\ document}{nombre\ de\ document\ avec\ le\ term\ w})$$

$$TFIDF(w) = TF(w) * IDF(w)$$

1.4.4. Le modèle LDA (Latent Dirichlet Allocation)

La LDA est utilisé pour classer le texte d'un document en fonction d'un sujet particulier. Il construit un sujet par modèle de document et des mots par modèle de sujet.
Chaque document est modélisé comme une distribution multinomiale de sujets et chaque sujet est modélisé comme une distribution multinomiale de mots.
La LDA suppose que chaque morceau de texte que nous y introduisons contiendra des mots qui sont d'une manière ou d'une autre liés. Il est donc crucial de choisir le bon corpus de données.
Elle suppose également que les documents sont produits à partir d'un mélange de sujets. Ces sujets génèrent ensuite des mots en fonction de leur distribution de probabilité.
Les différentes phases du modèle :

1.4.4.1. Extraction de thèmes à l'aide de LDA en Python

<u>Prétraitement du texte brut</u>

Cela implique ce qui suit :

- La symbolisation : Diviser le texte en phrases et les phrases en mots. Mettre les mots en minuscules et supprimez la ponctuation.
- Les mots qui comportent moins de 3 caractères sont supprimés.
- Tous les mots d'arrêt (stop words) sont supprimés.
- Les mots sont lemmatisés : les mots à la troisième personne sont changés en première personne et les verbes aux temps passés et futurs sont changés en présent.
- Les mots sont égrainés - ils sont réduits à leur forme racine.

1.4.4.2. Conversion du texte en sac de mots

Avant la modélisation du sujet, nous convertissons le texte symbolisé et lemmatisé en un sac de mots - que l'on peut considérer comme un dictionnaire où la clé est le mot et la valeur est le nombre de fois que ce mot apparaît dans l'ensemble du corpus.

1.4.4.3. Lancement du modèle LDA

C'est en fait assez simple puisque nous pouvons utiliser le modèle LDA de Gensim. Nous devons préciser le nombre de thèmes présents dans l'ensemble de données.

Détails de l'algorithme LDA en Annexe 7.1.

1.4.5. L'algorithme du Kmeans

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. Par ailleurs, une observation ne peut se retrouver que dans un cluster à la fois (exclusivité d'appartenance). Une même observation, ne pourra donc, appartenir à deux clusters différents.

Pour pouvoir regrouper un jeu de données en K cluster distincts, l'algorithme K-Means a besoin d'un moyen de comparer le degré de similarité entre les différentes observations. Ainsi, deux données qui se ressemblent, auront une distance de dissimilarité réduite, alors que deux objets différents auront une distance de séparation plus grande. Lors des essais avec KMeans, nous utiliserons la distance Euclidienne. Soit une matrice X à n variables quantitatives. Dans l'espace vectoriel E^n. La distance euclidienne d, entre deux observations x_1 et x_2 se calcule comme suit :

$$d(x_1, x_2) = \sqrt{\sum_{j=1}^{n} (x_{1n} - x_{2n})^2}$$

Choisir un nombre de cluster K n'est pas forcément intuitif. Spécialement quand le jeu de données est grand et qu'on n'ait pas un a priori ou des hypothèses sur les données. Un nombre K grand peut conduire à un partitionnement trop fragmenté des données. Ce qui empêchera de découvrir des patterns intéressants dans les données. Par contre, un nombre de clusters trop petit, conduira à avoir, potentiellement, des cluster trop généralistes contenant beaucoup de données. Dans ce cas, on n'aura pas de patterns "fins" à découvrir.

Pour un même jeu de données, il n'existe pas un unique clustering possible. La difficulté réside donc à choisir un nombre de cluster K qui permettra de mettre en lumière des patterns intéressants entre les données. Malheureusement il n'existe pas de procédé automatisé pour trouver le bon nombre de clusters.

K-Means en particulier et les algorithmes de clustering de façon générale ont tous un objectif commun : Regrouper des éléments similaires dans des clusters. Ces éléments peuvent être tout et n'importe quoi, du moment qu'ils sont encodés dans une matrice de données.

1.4.6. Word embeddings : Word2vec, Glove et FastText

En une phrase, les *word embeddings* sont des vecteurs qui représentent les mots d'un corpus (ici, un corpus est un ensemble de phrases). Tous les mots (même ceux qui ne sont dans aucun dictionnaire) peuvent être représentés sous forme de *word embeddings*. Il suffit seulement d'avoir un corpus de grande taille sous la main. Ils sont très utiles pour évaluer la parenté sémantique (plus simplement, la relation de sens) entre les mots. À titre d'exemples, voici quelques paires de mots sémantiquement parents:
Arbre / Forêt, Arbre / Branche, Arbre / Érable, Arbre / Plante
La proximité des vecteurs dans l'espace vectoriel correspond à la parenté sémantique: les vecteurs des mots se retrouvant dans des contextes similaires dans le corpus ont tendance à se rapprocher les uns des autres. La parenté sémantique entre deux mots peut alors être évaluée avec une mesure comme la similarité cosinus, la distance euclidienne ou la distance de Manhattan.

Il existe plusieurs modèles pour entraîner des *word embeddings*. Les plus connus sont Word2Vec, GloVe et FastText.

Word2Vec vient en deux saveurs:
· *Continuous bag-of-words*
· *Skip-gram*

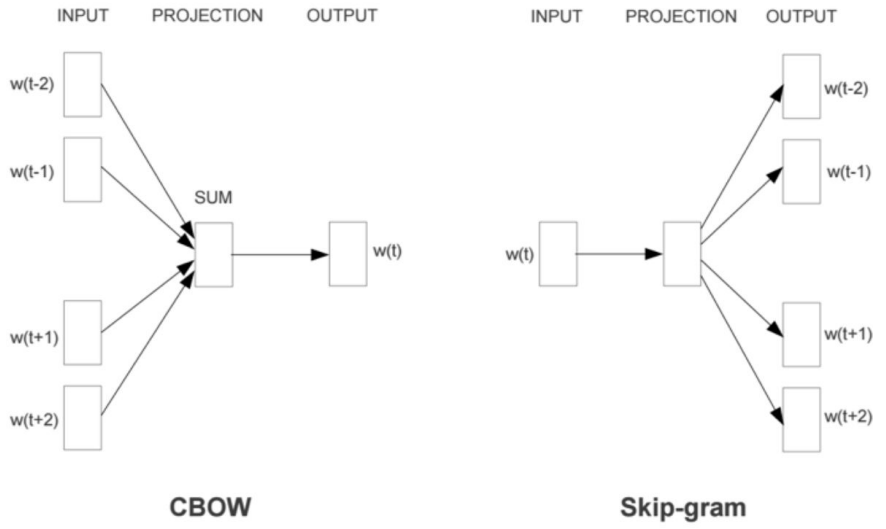


Figure 2 : https://arxiv.org/pdf/1301.3781.pdf

En fait, ces deux architectures correspondent à deux façons légèrement différentes d'entraîner les word embeddings. CBOW reçoit en entrée le contexte d'un mot, c'est à dire les termes qui l'entourent dans une phrase, et essaye de prédire le mot en question. Skip-Gram fait exactement le contraire : elle prend en entrée un mot et essaye de prédire son contexte. Dans les deux cas, l'entraînement du réseau se fait en parcourant le texte fourni et en modifiant les poids neuronaux afin de réduire l'erreur de prédiction de l'algorithme. Etant donné que Word2Vec n'est composé que de deux couches, cet algorithme est rapide à entraîner et à exécuter, ce qui se révèle être un avantage important par rapport à d'autre méthodes de word embedding.

Pennington et al. soutiennent que l'approche de balayage en ligne utilisée par word2vec est sous-optimale car elle n'exploite pas pleinement les informations statistiques globales concernant les cooccurrences de mots.
Dans le modèle qu'ils appellent Global Vectors (GloVe), ils disent: «Le modèle produit un espace vectoriel avec une sous-structure significative, comme en témoignent ses performances de 75% sur une tâche d'analogie de mots récente. Il surpasse également les modèles connexes sur les tâches de similitude et la reconnaissance des entités nommées. »
Afin de comprendre le fonctionnement de GloVe , nous devons comprendre deux méthodes principales sur lesquelles GloVe a été construit: la factorisation matricielle globale et la fenêtre de contexte local.
Voici comment cela fonctionne : au lieu d'extraire les plongements d'un réseau de neurones conçu pour effectuer une tâche différente comme prédire les mots voisins (CBOW) ou prédire le mot de focus (Skip-Gram), les plongements sont optimisés directement , de sorte que le le produit scalaire de deux vecteurs de mots est égal au logarithme du nombre de fois où les deux mots se produiront l'un près de l'autre.

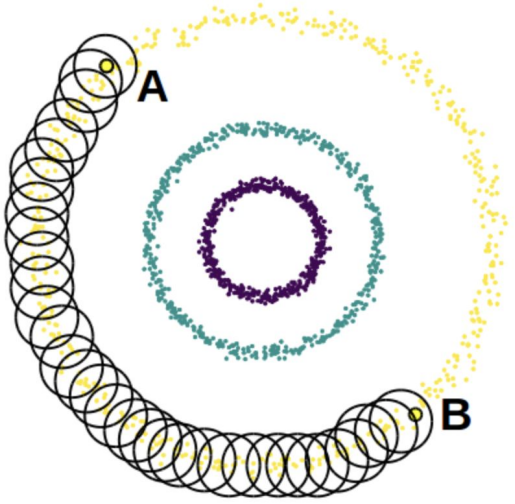
Détails sur l'algorithme de Glove en Annexe 7.4.

FastText est une autre méthode d'intégration de mots qui est une extension du modèle word2vec. Au lieu d'apprendre directement des vecteurs de mots, fastText représente chaque mot comme un n-gramme de caractères. Ainsi, par exemple , prenez le mot « artificiel » avec n = 3, la représentation fastText de ce mot est < ar, art, rti, tif, ifi, fic, ici, ial, al >, où les crochets angulaires indiquent le début et fin du mot .

Cela aide à saisir le sens des mots plus courts et permet aux plongements de comprendre les suffixes et les préfixes. Une fois que le mot a été représenté à l'aide de n-grammes de caractères, un modèle de saut de gramme est formé pour apprendre les plongements. Ce modèle est considéré comme un modèle de sac de mots avec une fenêtre coulissante sur un mot car aucune structure interne du mot n'est prise en compte. Tant que les caractères sont dans cette fenêtre, l'ordre des n-grammes n'a pas d'importance.

1.4.7. L'algorithme du DBSCAN

L'algorithme DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) a été introduit en 1996 dans ce but. Cet algorithme est très utilisé, raison pour laquelle il a obtenu en 2014 une distinction de contribution scientifique ayant résisté à l'épreuve du temps.



On peut connecter A à B par des petits voisinages (les cercles) contenant uniquement des points du cluster extérieur.

DBSCAN itère sur les points du jeu de données. Pour chacun des points qu'il analyse, il construit l'ensemble des points atteignables par densité depuis ce point : il calcule l'epsilon-voisinage de ce point, puis, si ce voisinage continent plus de n_min points, les epsilon-voisinages de chacun d'entre eux, et ainsi de suite, jusqu'à ne plus pouvoir agrandir le cluster. Si le point considéré n'est pas un point intérieur, c'est à dire qu'il n'a pas suffisamment de voisins, il sera alors étiqueté comme du bruit. Cela permet à DBSCAN d'être robuste aux données aberrantes puisque ce mécanisme les isole.

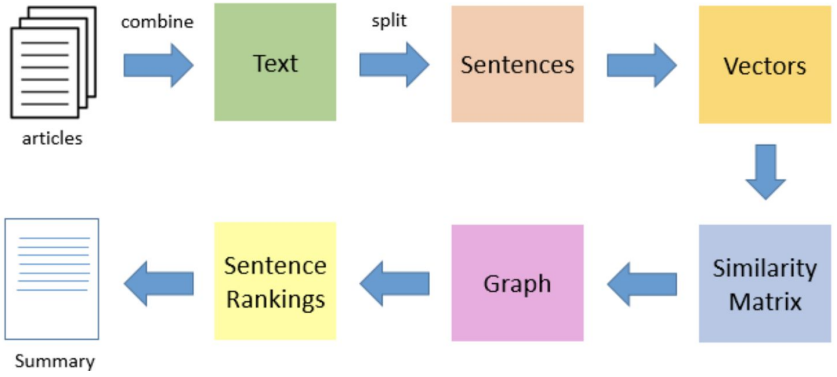
L'algorithme DBSCAN est difficile à utiliser en très grande dimension : à cause du fléau de la dimensionnalité. Les boules de rayon epsilon et de grande dimension ont tendance à ne contenir aucun autre point.
Le choix des paramètres ϵ et nmin peut aussi être délicat : il faut veiller à utiliser des paramètres qui permettent de créer suffisamment de points intérieurs (ce qui n'arrivera pas si nmin est trop grand ou ϵ trop petit). En particulier, cela signifie que DBSCAN ne pourra pas trouver des clusters de densités différentes.

DBSCAN a le grand avantage d'être efficace en temps de calcul sans requérir de prédéfinir le nombre de clusters.

1.4.8. L'algorithme de TextRank

TextRank est une technique de synthèse de texte extractive et non supervisée. Cette technique a été introduite par Rada Mihalcea and Paul Taurau dans le leur publication : *"TextRank: Bringing Order Into Texts"* en 2004.
Ce papier vise à introduire TextRank, un modèle de classement basé sur des graphiques qui peut être utilisés dans le traitement de texte.
TextRank prend ses racines dans l'algorithme de PageRank (*Voir Annexe 7.2.)* qui lui consiste à calculer le poids sur des pages d'internet. Cet algorithme est utilisé par Google Search afin de classer les pages web dans les résultats des moteurs de recherche.

TextRank est une technique qui permet donc l'extraction de phrases sur Python. Le but de cet algorithme est d'extraire les phrases les plus pertinentes d'un corpus de textes.
La figure ci-dessous nous explique le fonctionnement de TextRank:



La première étape consiste à concaténer le texte contenu dans les articles et puis diviser le texte en phrases individuelles. La deuxième étape consistera à trouver une représentation vectorielle (intégration de mots) pour chaque phrase. Les similitudes entre les vecteurs de phrases sont ensuite calculées et stockées dans une matrice. La matrice de similitude est ensuite convertie en un graphique, avec des phrases comme sommets et des scores de similitude comme arêtes, pour le calcul du rang de la phrase. Enfin, un certain nombre de phrases les mieux classées forment le résumé final.

2. Les méthodes utilisées pour la "text summarization"

2.1. La gestion des données: phase de pré processing

Nous avons débuté nos travaux en recherchant comment la machine était capable de résumer du texte. Nous avons observé que chacun des papiers sur ce sujet suivait une cheminement typique à savoir une phase de pré-processing (nettoyer le texte avant tout traitement) qui consistait en :

- Convertir tous les mots en minuscules, pour éviter que la machine ne pense que deux mots similaires aient des significations différentes.
- Supprimer les balises HTML
- Cartographie de la contraction et remplacer par les mots sans contraction.
- Supprimer ('s)
- Supprimer tout texte entre parenthèses ( ) pour éviter qu'il pense avoir affaire à une fonction ou autre.
- Éliminer les ponctuations et les caractères spéciaux
- Supprimer les mots-clés
- Supprimer les mots courts ou encore dits stop-words : qui sont des mots ayant peu de pertinence pour le sous-entendu des phrases. Ce sont des mots comme enfin, tous, de, à, vraiment, divers, juste, la plupart ...
  En fait, ils sont utiles en langue anglaise mais pas dans le traitement du langage naturel

Une fois que toutes ces étapes sont réalisées, on obtient un texte "nettoyé" et un document représente alors un enchaînement de mots sans ponctuation ni aucun caractères spéciaux etc.

Il est important de noter que les stops words ne sont pas directement supprimés dans cette partie du preprocessing, puisque ils sont aussi importants. Ils seront pris en compte dans chacune des étapes de la modélisation.
Étant donné que tous les documents utilisés ici concernent Airbus Helicopter, il est évident de trouver ces deux mots partout dans le document. A cet effet, nous avons choisi d'ajouter "airbus" et "helicopter" dans le dictionnaire des stop words. De même, certains mots tel que "from", "subject", "also", "take", "unmaned", …, qui ne présentent pas un intérêt particulier.

Chaque document nettoyé est ensuite transformé en liste de mots (un document une liste). Ces listes sont utilisées par les n-grammes et le modèle LDA.
On s'est alors intéressé aux étapes suivantes qui font partie du preprocessing à savoir :

Le stemming:
Le "stemming" est le processus de production de variantes morphologiques de la racine d'un mot ou encore appelée base. Cet algorithme de dérivation réduit les

mots, comme par exemple "chocolates", devient "choco" "retrieval", "retrieved", "retrieves" deviennent "retrieve".

Il y a principalement deux erreurs/problèmes dans le stemming - over stemming et under temming. Le 1er apparaît quand deux mots dont la racine est différente et qui ont un même sens sont associés
Le second se produit lorsque deux mots dont la racine est la même et qui ont un sens différent sont associés.

Cette méthode est principalement utilisée pour :

- Générer les systèmes de recherche d'informations comme les moteurs de recherche.
- Déterminer les vocabulaires dans l'analyse de domaine.

Les différents algorithmes de stemming que l'on retrouve dans la littérature sont les suivants :

- Algorithme de Stemmer de Potter

C'est l'une des méthodes de stemming les plus populaires proposées en 1980. Elle est basée sur l'idée que les suffixes en langue anglaise sont constitués d'une combinaison de suffixes plus petits et plus simples.

- Lovins Stemmer

Il est proposé par Lovins en 1968, qui supprime le suffixe le plus long d'un mot puis le mot est recodé pour convertir sa racine en mots valides.

- N-Gram Stemmer

Un n-gram est un ensemble de n caractères consécutifs extraits d'un mot dans lequel des mots similaires auront une forte proportion de n-grammes en commun.
etc.

Les inconvénients sont que toutes les formes infléchies doivent être explicitement mentionnées et la matrice des mots peut être volumineuse. Pour les langues à la morphologie simple, comme l'anglais, la taille des tableaux reste assez modeste, mais les langues à forte inflexion, peuvent avoir des centaines de formes infléchies potentielles pour chaque racine.

C'est la raison pour laquelle nous avons plus souvent recours à la lemmatisation.

La Lemmatisation:

La lemmatisation regroupe différentes formes infléchies d'un mot, appelé Lemma.
Elle est en quelque sorte similaire au terme "Stemming" car elle permet de faire correspondre plusieurs mots à une racine commune.
Le résultat de la lemmatisation est un mot propre :
Par exemple, gone, going, went donnent GO

Ensuite, une fois cette étape achevée, il est souvent utile de tokenizer un document. Le "Tokenizing" est le processus consistant à marquer ou à diviser une chaîne, un texte en une liste de tokens. On peut considérer les mots (= token) comme des parties d'une phrase, et une phrase comme le token d'un paragraphe.

Une fois cette étape effectuée, nous avons un texte brut et nous allons pouvoir construire un modèle permettant de vectoriser l'ensemble des mots. C'est à partir de cette étape que les méthodes diverges. Dans nos premières recherches, nous avons pu voir qu'il existait différentes méthodes pour résumer des documents.

2.2. Le topic modeling

2.2.1. Le topic modeling à travers le modèle de LDA

On peut être amener à exploiter plusieurs documents provenant d'origine divers ou non. Certains de ces documents peuvent porter des informations très similaires. Ainsi, il sera très intéressant d'identifier et de classer les textes qui véhiculent les mêmes informations ou qui portent les mêmes thèmes ou « topics ».

« Topic Modeling » est une technique qui permet d'extraire des topics dans un ou plusieurs textes. La difficulté majeure est de savoir comment extraire des topics de bonne qualité qui soient clairs, séparés et significatifs.

Dans cette partie, l'algorithme Latent Dirichlet Allocation (LDA) de modélisation de la librairie Gensim de Python sera utilisée pour faciliter cette tâche. Plusieurs méthodes, particulières de visualisations, seront utilisées pour évaluer la qualité de notre modèle.

L'approche de la LDA est de considérer chaque document comme un ensemble de collection de mots-clés, dans une certaine proportion. LDA permet donc de réorganiser la distribution des topics dans un document et la distribution des mots dans un topic.

Ce dernier utilise les modèles Online Latent Dirichlet Allocation (OLDA) comme présenté par Hoffman et al. dans leur papier intitulé « *Online Learning for Latent Dirichlet Allocation* ».

« Latent Dirichlet Allocation (LDA) is a Bayesian probabilistic model of text documents. It assumes a collection of K "topics". Each topic defines a multinomial distribution over the vocabulary and is assumed to have been drawn from a Dirichlet, βk ~ Dirichlet(η). Given the topics, LDA assumes the following generative process for each document d. First, draw a distribution over topics θd ~ Dirichlet(α). Then, for each word i in the document, draw a topic index zdi ∈ {1, . . . , K } from the topic weights zdi ~θd and draw the observed word wdi from the selected topic, wdi ~ βzdi . For simplicity, we assume symmetric priors on θ and β, but this assumption is easy to relax. »

2.2.2. Phrases Models: Bigram and Trigram model

Les modèles Phrases permettent une détection automatique des expressions courantes ou expressions à mots multiples encore appelé **n-grammes** (bigrammes, trigrammes, quadgrammes, …). Ces méthodes sont efficaces pour l'apprentissage des représentations vectorielles distribuées de haute qualité qui capturent un grand nombre de relations syntaxiques et sémantiques précises entre les mots. Dans notre exemple, nous nous limitons à utiliser les bigrammes et les trigrammes.

- Les bigrammes sont deux mots qui apparaissent fréquemment ensemble dans le document.
- De la même manière, les trigrammes sont trois mots qui apparaissent fréquemment.

Ainsi, avec cette méthode, tous les mots et bigrammes dont le nombre total collecté est inférieur à 5, fixée par le paramètre « min_count », sont ignoré. De même, un seuil de score pour la formation des phrases est fixé par l'argument « threshold » à 70. Autrement dit, une phrase de mots a suivi de b est acceptée si le score de la phrase est supérieur à 70. Ces deux valeurs sont choisies en se basant sur le score de cohérence du modèle LDA.

Voir Annexe 7.5. pour plus de détails sur N-gram

2.2.3. La mise en place du topic modelling

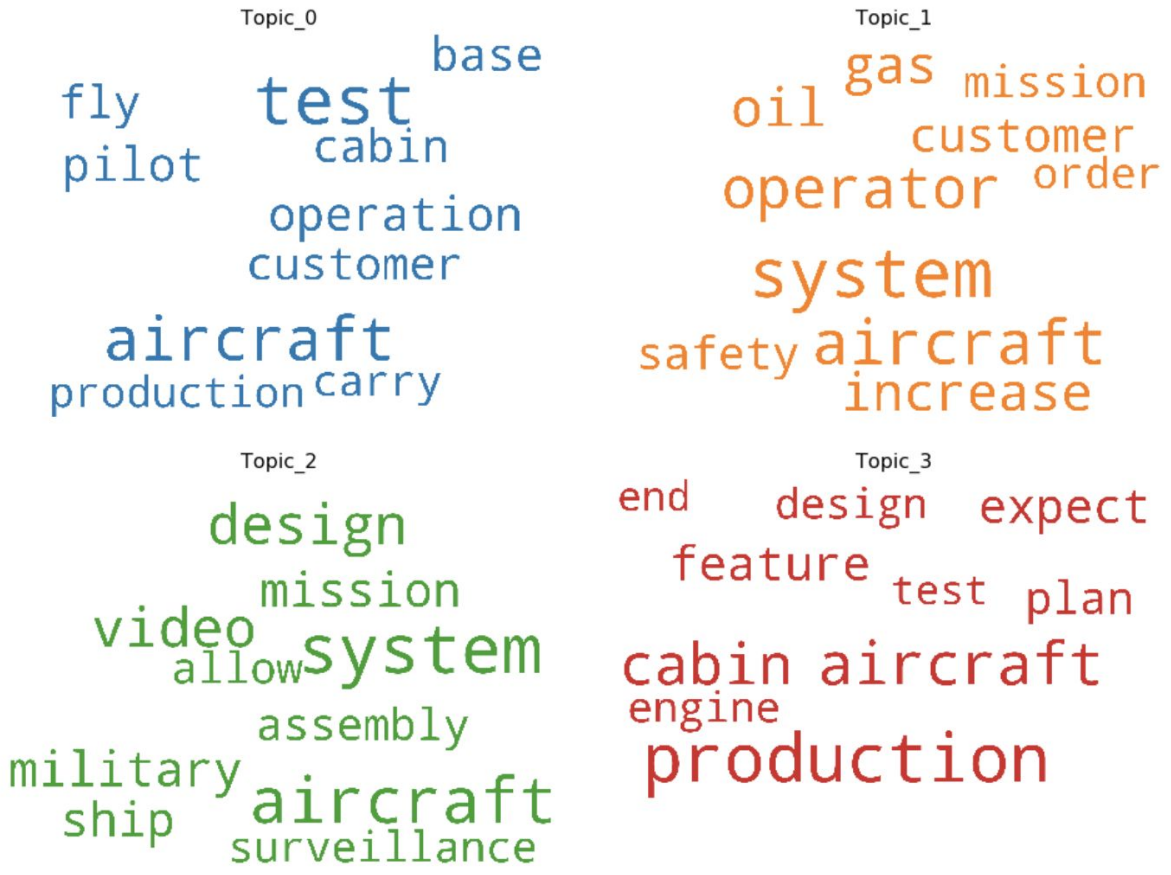
Le résultat des bigrammes et trigrammes est utilisé pour construire un dictionnaire (id2word) et un corpus à l'aide de l'algorithme Corpora de la librairie Gensim afin de mettre en place le modèle LDA.

En plus du corpus et du dictionnaire, le nombre de topic est initialement fixé à 4 et ensuite à 16. Le choix de cette dernière valeur est basé sur le score de cohérence du modèle. Le paramètre « chunksize » est le nombre de morceau de document utilisé l'entraînement du modèle.

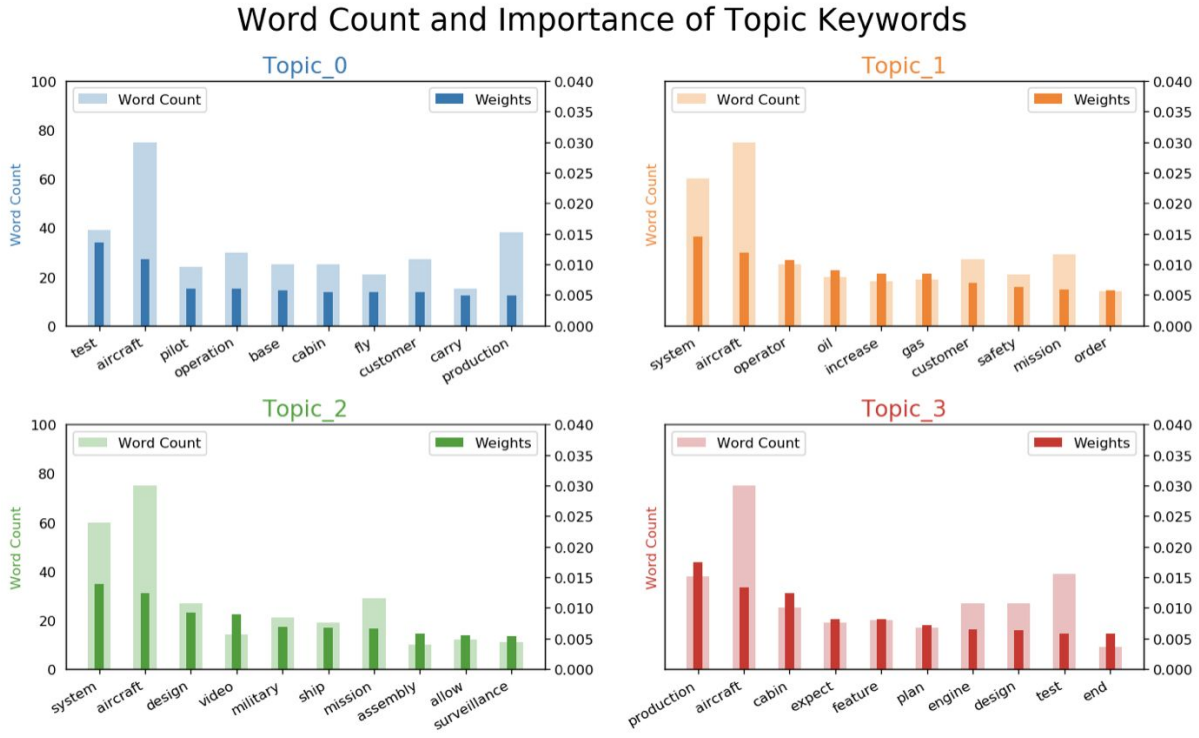
Le modèle LDA initial est entrainé avec 4 différents topics. Chaque topic représente une combinaison de mots et chaque mot contribue à un certain poids dans le topic. La fonction « lda_model.print_topics() » nous permet de voir les dix (10) premiers mots-clés et leur importance dans le chaque topic comme le montre ci-dessous.

```
[(0,
  '0.014*"test" + 0.011*"aircraft" + 0.006*"pilot" + 0.006*"operation" + '
  '0.006*"base" + 0.006*"cabin" + 0.005*"fly" + 0.005*"customer" + '
  '0.005*"carry" + 0.005*"production"'),
 (1,
  '0.015*"system" + 0.012*"aircraft" + 0.011*"operator" + 0.009*"oil" + '
  '0.009*"increase" + 0.009*"gas" + 0.007*"customer" + 0.006*"safety" + '
  '0.006*"mission" + 0.006*"order"'),
 (2,
  '0.014*"system" + 0.012*"aircraft" + 0.009*"design" + 0.009*"video" + '
  '0.007*"military" + 0.007*"ship" + 0.007*"mission" + 0.006*"assembly" + '
  '0.006*"allow" + 0.005*"surveillance"'),
 (3,
  '0.018*"production" + 0.013*"aircraft" + 0.012*"cabin" + 0.008*"expect" + '
  '0.008*"feature" + 0.007*"plan" + 0.006*"engine" + 0.006*"design" + '
  '0.006*"test" + 0.006*"end"')]
```

Par exemple, le mot le plus important dans le topic 0 est "test" avec un poids de 0.014 et le deuxième mot le plus important dans tous les quatre topics est "aircraft" avec un poids de 0.011 dans le topic 0, 0.012 dans les topics 1 et 2 et 0.013 dans le topic 3. Ci-dessous est une représentation simplifiée des dix premiers mots de chaque topic.



De la même façon, on peut compter et représenter le nombre de mot et le poids de chaque mot de chaque topic dans un graphe, comme suit.



Quel est le topic qui domine et quel est son pourcentage de contribution dans chaque document ? Ou encore, quel est le document le plus représenté dans chaque topic ?

Étant donné que les topics sont construits à partir des mots, chaque document peut être composé de plusieurs topics. Cependant, seul un des topics est dominant. Le tableau ci-dessous montre le topic dominant dans chaque document et son pourcentage de contribution. Dans le tableau, on voit clairement que, par exemple, 99,8% du contenu des documents numéro 25 et 0 appartiennent au topic 1.

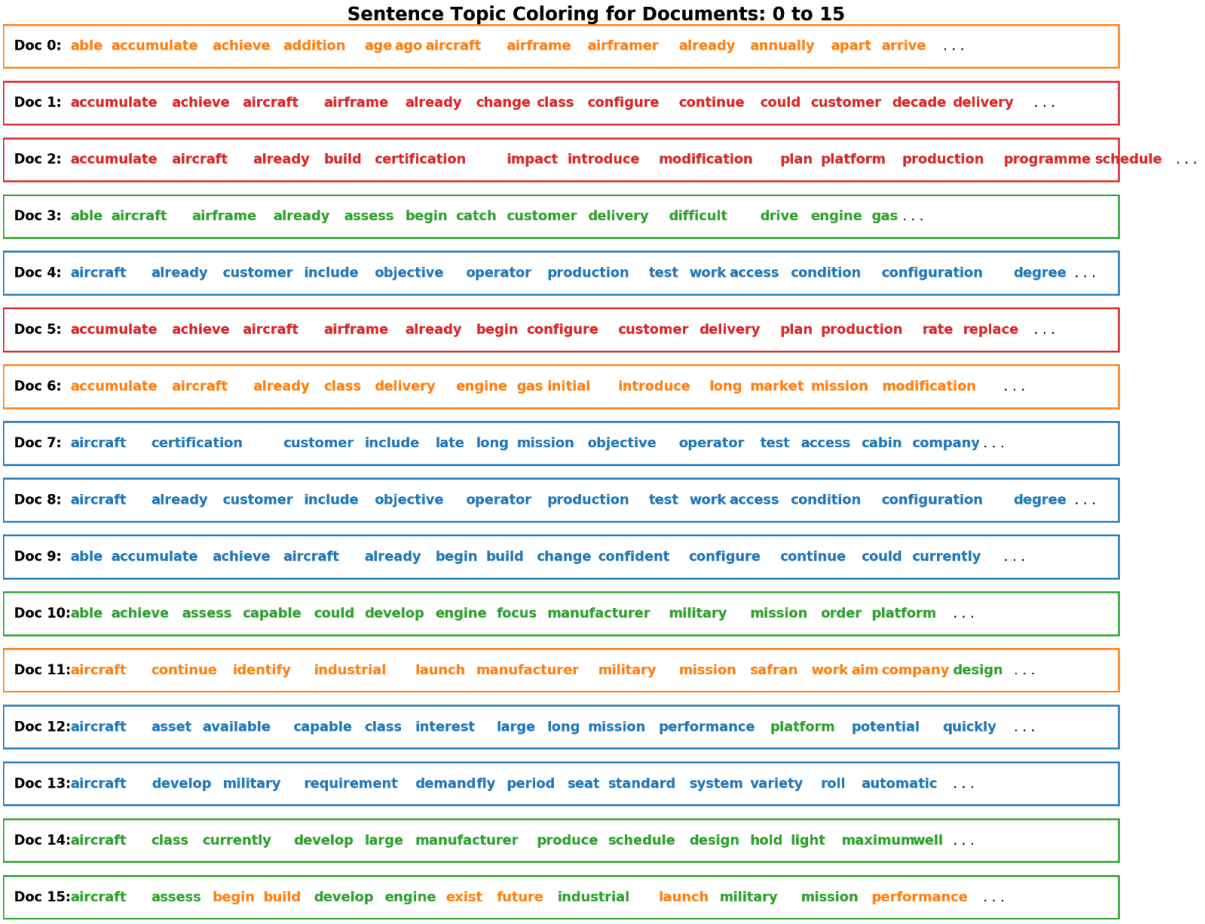
	doc_id	dominant_topic	topic_perc_contrib	keywords	text
25	25	1.0	0.9984	system, aircraft, operator, oil, increase, gas, customer, safety, mission, order	[celebrate, heritage, continue, safety, core, mission, issue, plate, shortage, approval, single,...
0	0	1.0	0.9981	system, aircraft, operator, oil, increase, gas, customer, safety, mission, order	[continue, weakness, oil, gas, market, leave, uncertain, quickly, able, ramp, output, airframer,...
3	3	2.0	0.9979	system, aircraft, design, video, military, ship, mission, assembly, allow, surveillance	[much, innovation, report, model, next, generation, apply, way, aircraft, well, design, file, pa...
23	23	3.0	0.9979	production, aircraft, cabin, expect, feature, plan, engine, design, test, end	[industry, gather, big, western, oem, schedule, bring, long, await, civil, model, market, promis...
9	9	0.0	0.9971	test, aircraft, pilot, operation, base, cabin, fly, customer, carry, production	[hope, significant, inroad, niche, market, marketing, director, booth, display, full, size, mock...

Inversement, on peut aussi vouloir présenter le document le plus représentatif dans chaque topic. En effet, dans un topic, tous les documents non pas le même poids. Il sera ainsi très intéressant de savoir quel document pèse le plus dans un topic donné. Les détails de cette partie sont disponibles sur le notebook.

Un récapitulatif est proposé dans le graphe ci-dessous. Il s'agit uniquement des 16 premiers documents. Dans ce graphe, chaque couleur correspond à un topic. Ainsi, on voit clairement que les documents qui portent la même couleur ont tendance à avoir la même composition de mots ou des mots qui sont très proches.

En outre, dans un même document, on peut avoir plusieurs topics. C'est exactement ce qu'on voit dans le document numéro 15. Il porte les topics 1 et 2. Cependant, la probabilité qu'un document donnée appartient à plusieurs topics est très faible, dans l'ensemble des documents. De même, pour les documents qui appartiennent à plusieurs topics, le pourcentage de contribution du topic dominant est élevé.

De ce fait, dans la suite de notre procédure, chaque document sera classé dans un seul topic. En d'autres termes, chaque document sera associé à son topic dominant.

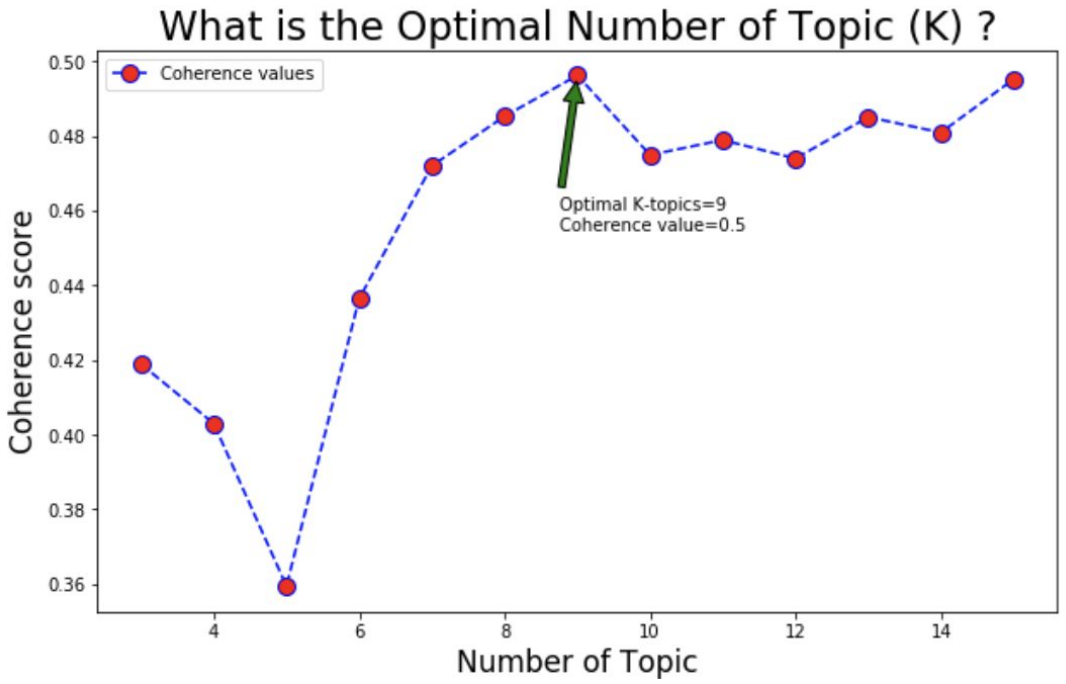


Comme les documents sont classés par topic, on peut maintenant penser à renommer les différents topics identifiés. En se basant sur les dix ou trente premiers mots les plus fréquents/importants, on peut donner un nom à chaque topic. Pour faire simple, nous avons choisi de retenir les trois premiers mots (par ordre d'importance) de chaque topic comme intitulé. Le tableau suivant montre le numéro du topic, les nombre de document dans chaque et les trois premiers mots les plus importants.

	topic_id	#_documents	3top_words
0	0	10	test ; aircraft ; pilot
1	1	6	system ; aircraft ; operator
2	2	8	system ; aircraft ; design
3	3	5	production ; aircraft ; cabin

2.2.4 Le nombre de topic optimal

Depuis le début de cette partie, nous avons imposé les nombre de topic à quatre pour des raisons de visualisation, de rapidité du traitement. On peut être amené à choisir de façon arbitraire le nombre de thème que l'on souhaite dans un lot de documents. Cependant, dans certains cas, ce nombre doit être en fonction de la nature des documents. Certains peuvent aller ensemble et d'autres non pas. Ainsi, pour résoudre ce problème, nous allons nous baser sur le score de cohérence de plusieurs topics. Il s'agit, donc, de proposer à notre modèle "LDA" construit précédemment plusieurs nombre de topics que nous allons nommé K-topics.

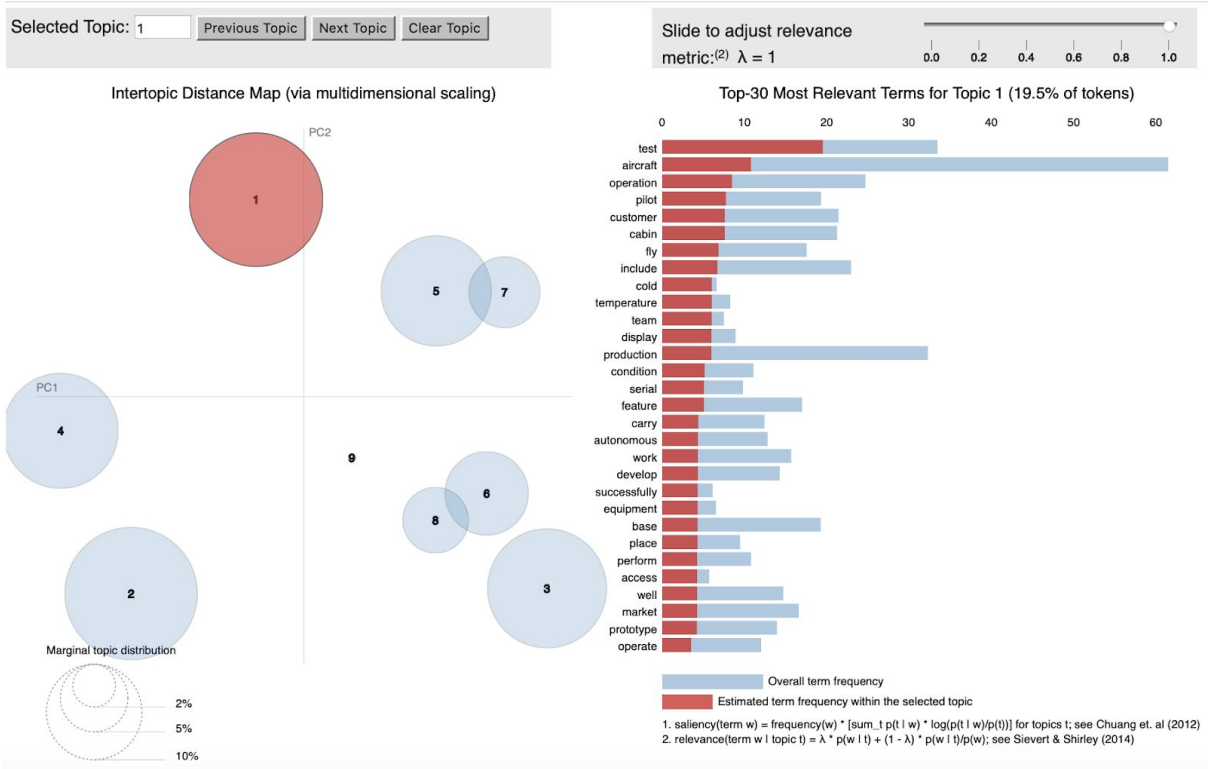


Pour chaque K-topics, le score de cohérence issu du modèle LDA sera stocké dans une fonction. Le nombre de topics retenu est celui qui aura le meilleur score.

Le graphe illustrant le **nombre optimal de topics** montre le score de cohérence du modèle LDA en fonction du nombre de topic (k-topics). Dans ce graphe, on constate que le score de cohérence est plus important lorsqu'on réparti nos documents sur 9 topics.

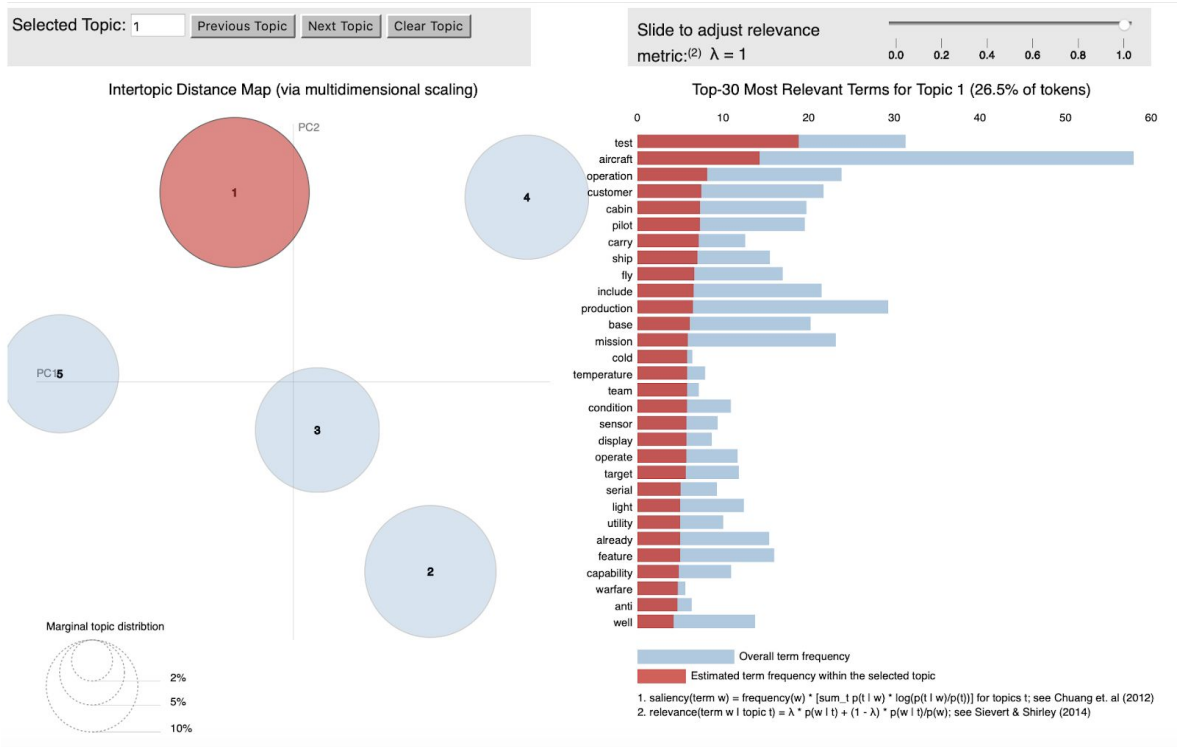
Néanmoins, la pertinence de cette méthode reste limitée du fait qu'elle ne tient pas en compte de la distance les informations entre les topics.

Intertopic Distance Map avec les 9-topics optimaux



Le graphe “Intertopic distance Map” montre que le topic numéro 9 ne contient pas d’informations importantes. De même, d’après la théorie des “Principal Component Analysis (PCA)”, les topics numéro 5 et 7 ont des informations similaires. Les topics 6 et 8 portent aussi des informations similaires. Ainsi, on peut cumuler les topics 5 et 7 et les topics 6, 8 et 9. Ce qui nous donnera au final 5 topics en tout.

Intertopic distance Map avec les 5-topics optimaux



Comme vous pouvez le constater, à gauche on les cinq topics optimaux. La distance entre ces topics est basée sur la proximité des informations contenues dans chaque topic. A droite, on les 30 premiers mots les plus importants du topic sélectionné (topic 1). Dans cette représentation, on voir directement la fréquence de chaque mots (en positionnant la souris sur le mot) dans chaque topic. Par exemple, lors qu'on se position sur le mot "cabin" le diamètre du topic 5 devient très important et les topics 2, 3, 4 disparaissent complètement. Cela équivaut à dire que ce mot "cabin" est présent uniquement dans les topics 1 et 5 et qu'il est beaucoup plus fréquent dans le dernier topic.

2.2.5 Regroupement des documents par topic

Au final, les 29 documents sont répartis entre les 5 topics. Par exemple, comme le montre le tableau ci-dessous, les documents 9, 12 et 18 sont classés dans le topic 0 avec des taux de contributions de 90%, 97% et 99% respectivement.

	doc_id	weigh_topic_4	weigh_topic_3	weigh_topic_2	weigh_topic_0	weigh_topic_1	topic_id	document
5	9	NaN	0.090091	NaN	0.907687	NaN	0	airbus helicopters is hoping to make significa...
6	12	NaN	NaN	0.027028	0.970927	NaN	0	aircraft vsr700 landing the vsr700 is airbus t...
7	18	NaN	NaN	NaN	0.996608	NaN	0	what is going on beyond the horizon it s a cru...
8	25	NaN	NaN	NaN	NaN	0.998300	1	hai is coming into this year s heli expo celeb...

On peut dès à présent regrouper tous les documents qui appartiennent au même topic en ensemble, comme suit:

- topic 0; 8 documents: document 28, 22, 4, 7, 8, 9, 12 et 18
- topic 1; 5 documents: document 25, 21, 6, 11 et 15
- topic 2; 5 documents: document 26, 20, 14, 3 et 27
- topic 3; 4 documents: document 5, 23, 2, et 1
- topic 4; 7 documents: document 16, 17, 13, 19, 10, 24 et 0

Ce regroupement est résumé dans la table suivante; où *topic_id* est l'identifiant unique du topic, *3top_words* les trois premier mots les plus importants dans le topic, *#_doc* le nombre de document dans chaque topic et *topic* l'ensemble des documents représentant le même topic.

topic_id	3top_words	#_doc	topic
0	test ; aircraft ; operation	8	a leader in helicopter customizations will dis...
1	system ; drone ; aircraft	5	hai is coming into this year s heli expo celeb...
2	aircraft ; video ; system	5	imt vislink booth c4338 is showcasing its upda...
3	production ; aircraft ; cabin	4	the first serial production airbus helicopters...
4	gas ; oil ; military	7	the airbus helicopters vsr700 demonstrator a l...

2.2.5. Topic Modeling par groupement de texte

Une dernière technique consiste à vectoriser les documents et à effectuer un topic modeling entre eux. Chaque phrase de ces documents sera classée dans ces topics. Le but est de gagner en cohérence en cherchant à résumer des groupes de textes dans la partie summarization plutôt que des thèmes abstraits communs à des phrases de plusieurs textes différents.

Afin de faire ressortir le nombre optimal de thèmes nous avons effectué un grid search sur les paramètres clés de l'algorithme DBSCAN (epsilon et nombre minimal par coeur de cluster) et le nombre de clusters de K-means. Nous avons utilisé le score de silhouette afin de départager les différents résultats.

L'algorithme de DBSCAN et k-means ont trouvé un nombre optimal de cluster égal mais nous avons sélectionné k-means pour sa simplicité et sa vitesse de convergence.
Une fois le nombre de cluster choisi, nous avons utilisé un modèle LDA afin de détecter les mots composant les topics du corpus. Une fois cela fait, nous avons choisi de résumer le nombre de topic que nous avons détecté auparavant. Nous nommons les topics par le mot le plus pertinent détecté par le LDA.

2.3. L'extraction des phrases

2.3.1. Pytext rank

Comme nous avons pu voir dans la première partie, Pytext Rank permet de faire synthèses de textes de manière extractive et non supervisée.

Dans un premier temps on examine les résultats. On crée une liste des phrases les mieux classées dans le document.

```
Entrée [71]: for p in doc._.phrases:
                 print("{:.4f} {:5d}  {}".format(p.rank, p.count, p.text))
                 print(p.chunks)

0.0775     2  airbus helicopters flight control system
[airbus helicopters flight control system, airbus helicopters flight control system]
0.0741     1  airbus helicopters automatic flight control system
[airbus helicopters automatic flight control system]
0.0667     1  airbus helicopters site
[airbus helicopters site]
0.0665     3  airbus h145 helicopters
[airbus h145 helicopters, airbus h145 helicopters, airbus h145 helicopters]
0.0659     1  airbus corporate helicopters
[airbus corporate helicopters]
0.0658     2  manned helicopter flights
[manned helicopter flights, manned helicopter flights]
0.0656     1  airbusthe airbus helicopters
[airbusthe airbus helicopters]
0.0656     1  airbus helicopters sas
[airbus helicopters sas]
0.0620     1  airbus helicopters marketing director
[airbus helicopters marketing director]
0.0617     1  airbus helicopters unique expertise
```

On crée ensuite une liste qui comprend des vecteurs délimiteurs de phrases. Par exemple [0,31], est la première phrase qui va du caractère 0 au caractère 31 de l'ensemble du texte.
Ensuite, on ajoute l'étape précédente au vecteur créé avec les phrases les mieux classées.

```
0 6 airbus helicopters flight control system 0.07754052398700356
  5588 5593
  5569 5588 5593 5612
  8145 8150
  8126 8145 8150 8169
1 6 airbus helicopters automatic flight control system 0.07411850373726805
  5625 5631
  5612 5625 5631 5646
2 6 airbus helicopters site 0.06674427211808748
  13681 13684
  13635 13681 13684 13690
3 6 airbus h145 helicopters 0.06647065341740242
  3378 3381
  3365 3378 3381 3391
  4290 4293
  4277 4290 4293 4318
  4562 4565
  4549 4562 4565 4575
4 6 airbus corporate helicopters 0.06589264506183609
  1818 1821
  1814 1818 1821 1833
5 6 manned helicopter flights 0.0658469808398924
  7928 7931
  7913 7928 7931 7935
  9250 9253
  9235 9250 9253 9257
```

Puis on crée un “pré-résumé qui consiste à récupérer les 10 phrases les mieux classées de chaque document (ici nous avons choisi 10 phrases, mais on peut choisir un autre nombre). Nous construisons également un vecteur_unitaire pour toutes les phrases, jusqu'à la limite demandée. Puis, on itère chaque phrase en calculant sa distance euclidienne par rapport au vecteur unitaire. Et enfin, on extrait les phrases avec la distance la plus faible, jusqu'à la limite demandée.

2.3.2. Méthode extractive

a) Lemmatize words

Étant donné que la méthode est basée sur la fréquence des mots, il est très intéressant de tenir en compte de la proximité des mots du document (ou de chaque topic). Un même mot peut être utilisé, dans une phrase donnée, au pluriel ou féminin alors dans non dans une autre phrase. Dans ce contexte, un algorithme classique qui ne tient ne tient pas compte de cette complexité identifiera plusieurs fois le même mot. La fonction “Lemmatize” disponible dans la librairie “nltk” permet de pallier cette difficulté. Cette fonction “Lemmatization” consiste généralement à faire les choses correctement en utilisant un vocabulaire et une analyse morphologique des mots, visant normalement à supprimer les terminaisons inflexibles et à rendre la forme de base. Ci-dessous est un exemple utilisant quelques mots.

```
w = "caresses ponies caress cats dogs"
lemmatize_words(word_tokenize(w))
```

```
['caress', 'pony', 'caress', 'cat', 'dog']
```

b) La fréquence de chaque mot dans un document

Comme son nom l'indique, la fonction construite manuellement permet, dans document donné, de trouver les mots les plus courants ou les moins courants. Concrètement, il s'agit de compter le nombre de fois que chaque mot apparaît dans le document. Elle remet tous les mots en minuscule, ignore les stop words (et les mots comme airbus, helicopter, also, …) et les mots dont la longueur est inférieure ou égale à deux (mots avec 1 ou 2 lettres).

```
t = "The dogs are nice. The dogs are out. I like dogs."
freq(word_tokenize(t))
```

```
{'dogs': 3, 'like': 1}
```

c) Le score des phrases

Comme le score attribué à chaque phrase détermine l'importance de la phrase, il devient extrêmement important de choisir le bon algorithme pour trouver ce score. Notre approche utilise le score TF-IDF (avec la fonction "word_tfidf")de chaque mot pour calculer le score total de la phrase en utilisant les résultats des fonctions "pos_tagging", et "lemmatize_words".

d) POS tagging

Basé sur la bibliothèque nltk, cette fonction fait le tour du document pour analyser tous les mots du texte et ne retourne que les **noms** et les **verbes** du texte.

```
The : DT
dogs : NNS
are : VBP
nice. : IN
The : DT
dogs : NNS
are : VBP
out. : JJ
I : PRP
like : VBP
dogs. : NNS
```

3. Les résultats

3.1. Méthode Extractive (Pytextrank)

La méthode extractive fait avec PYTEXTRANK permet donc d'extraire un certain nombre de phrases. Pour rappel, c'est à nous de choisir le nombre de phrases que nous souhaitons pour le résultat.
Dans ce cas là, nous avons choisi de garder 10 phrases lors du résumé. PytextRank sélectionne les 10 phrases les mieux classées.

```
243 passing this first step of autonomous flights with a safety pilot onboard allows us to validate the integration
of airbus helicopters flight control system with the aerial vehicle and its specific engine installation said bruno
guimbal , president ceo of helicopteres guimbal.
345 passing this first step of autonomous flights with a safety pilot onboard allows us to validate the integration
of airbus helicopters flight control system with the aerial vehicle and its specific engine installation , said hel
icopteres guimbal president and ceo bruno guimbal.
244 this phase of flight trials with a safety pilot will focus on refining airbus helicopters automatic flight cont
rol system aboard the opv , eventually leading to fully autonomous flights without a safety pilot.
580 including search and rescue operations , narcotics surveillance interdiction , personnel cargo transport , inte
r agency pursuit coordination , wildfire suppression , emergency medical services transport , and patrol support.th
e second prototype of the five bladed h145 helicopter carried out its maiden flight in august from airbus helicopte
rs site in donauw rth , germany.
153 operations took place from the base of babcock scandinavian airambulance , which flies airbus h145 helicopters
for emergency medical services ems customers in the region.
199 operations took place from the base of babcock scandinavian airambulance , which flies airbus h145 helicopters
for emergency medical services ems customers in the region , including finland s finnhems and the regional governme
nt of aland , also in finland.
208 operations took place from the base of babcock scandinavian airambulance , which flies airbus h145 helicopters
for emergency medical services ems customers in the region.
69 earlier this year , airbus corporate helicopters ach announced the sale of an ach160 to a european customer.
336 the system will initially offer extended surveillance capabilities for navies , allowing them to preserve manne
d helicopter flights for critical missions.
397 the system will initially offer extended surveillance capabilities for navies , allowing them to preserve manne
d helicopter flights for critical missions.
```

Au début de chaque phrase sélectionnée on peut voir le numéro de la phrase. Il faut savoir que dans notre cas, les 29 documents ont 595 phrases au total. Dans le résumé, on voit que PytextRank sélectionne des phrases à différents endroits des documents. On remarque aussi que les 2 premières phrases du résumé sont très semblables. Il n'y a que la fin qui diffère. Cela montre que l'algorithme est efficace car ce sont les 2 phrases les mieux classées.

Concernant le sens des phrases, on voit donc que les 2 premières phrases parlent des essais de vol. Les phrases au milieu du résumé traite de l'hélicoptère h145. Puis il y a 2 phrases sur l'hélicoptère h160 et enfin le résumé se termine par un bilan sur les ventes d'Airbus Helicopter.

3.2. Méthode Extractive

3.2.1. Récupérer / trouver les phrases les plus importantes

Pour trouver les phrases les plus importantes, on utilise les phrases individuelles issue de la fonction "tokenized" et on calcule le score de la phrase.

Après le calcul des scores, les phrases les plus importantes, en fonction du taux de rétention (pourcentage de phrases dans le topic) fourni par l'utilisateur, sont incluses dans le résumé. Par exemple, avec un document (ou topic) de 120 phrases, si l'utilisateur demande 10% du texte, les 12 phrases les plus importantes sont imprimées. Dans l'image suivante, vous trouverez les 5% des phrases les plus importantes dans chacun des 5 topics.

```
Percentage of information to retain(%): 5
Summarize topic_0 in 8 sentences ==============================>:
6 inch horizontal format can accommodate primary flight display pfd information and a multifunction display mfd side by side within the same unit. There are odd ones for the national guard , combat training center , national training center. The h160 is particularly well suited for long distance hems missions because of its 150 knot cruise speed , low cabin vibration , robust cabin air conditioning , flat approach angle , easy loading and unloading , ample artificial and natural cabin lighting , and generous cabin volume that facilitates 360 degree patient access , he said. Anti submarine warfare asw vsr700 anti submarine warfare the vsr700 is capable of carrying and positioning an anti submarine warfare asw barrier to complement a ship s asw activity or that of a helicopter with manned unmanned teaming mumt. Anti surface warfare asuw vsr700 anti surface anti surface warfare asuw is conducted in very hostile environments where a potential threat has to be distinguished from neutral or friendly sea traffic. Maritime security vsr700 maritime security in maritime security missions , the ability to scan a large maritime zone , track the activity within it , identify contacts of interest and remain discreet is essential. Search and rescue sar vsr700 search and rescue search and rescue sar is often a race against time.
Summarize topic_1 in 6 sentences ==============================>:
Hai returns to heli expo after capping an intense lobbying campaign to combat user fees and secure a long range faa reauthorization bill. The forum tends to draw a standing room only crowd. Data proves customers experience an up to 18 percent increase in useful load and a 10 percent increase in available right pedal margin at high power settings , which translates into a 100 percent payback on the system investment in as little as 21 months. The fastfin system is approved by the federal aviation administration faa , transport canada tc , the european aviation safety agency easa , and numerous other civil aviation authorities around the world. H160 timeline 2011 the h160 is first revealed as the x4 concept march 2015 the x4 was renamed the h160 june 2015 the first h160 prototype takes its first test flight december 2015 the second prototype performed its initial ground run january 2016 the second prototype took its maiden flight october 2017 third h160 prototype revealed march 2018 babcock announced as launch customer march 2018 airbus sells the first ach160 , the airbus corporate version.
Summarize topic_2 in 7 sentences ==============================>:
Imt vislink booth c4338 is showcasing its updated airborne video downlink system avds with the addition of the tsm 2020 transport management system at the hai heli expo 2019. imt vislink s avds is an integrated suite of downlink transmitters , receivers and antennas that creates a comprehensive aerial based video transmission solution. Including search and rescue operations , narcotics surveillance interdiction , personnel cargo transport , inter agency pursuit coordination , wildfire suppression , emergency medical services transport , and patrol support. It includes a 4 axis autopilot , increasing safety and reducing pilot workload. We used a lot of methodology coming from the automotive sector such as ring methodology. The 169 hils will replace the dauphin , gazelle , dauphin , fennec and alouette iii in the three french armed forces.
Summarize topic_3 in 4 sentences ==============================>:
Program milestones last year included successful drop testing of the production landing gear , production engine certification testing on ac3 , continued fuselage and empennage fatigue testing , and installation of a new production main cabin door designed with an embedded rescue hoist for search and rescue operations. Airbus helicopters announced the h160 fuselage tail boom join up in july. Ach managing director frederic lemos said , it is particularly pleasing to see the new ach160 continuing to win orders.
Summarize topic_4 in 6 sentences ==============================>:
The uav was piloted and monitored from the ground station located at the base. The aircraft was piloted and monitored from a ground station located at the base. Passing this first step of autonomous flights with a safety pilot onboard allows us to validate the integration of airbus helicopters flight control system with the aerial vehicle and its specific engine installation said bruno guimbal , president ceo of helicopteres guimbal. Newly signed up customers include dare county north carolina medflight , the ukrainian ministry of interior , advance flight of new zealand , the norwegian air ambulance services , and swiss air rega. On the support side , airbus signed contracts covering the 19 helicopters in the uk s national police air service fleet.
```

3.2.2. Dans quel document est extraite la phrase numéro *n* du topic *k* ?

Pour répondre à cette question, Nous avons établi une fonction en se basant sur les phrases en sortie et les phrases du document d'origine. Notre méthodologie consiste à rechercher chaque phase extraite (output) dans l'ensemble des documents originaux. Une fois la phrase extraite retrouvée, on récupère le numéro du document correspondant et le numéro de cette même phrase dans ce document. Le tableau suivant montre, respectivement en colonne, le numéro du document, le numéro de la phrase dans le document, le numéro du topic, le numéro de la phrase dans le topic et la phrase extraite à partir du topic. Il s'agit uniquement des 5% des phrases les plus importantes dans chaque topic.

	DocumentID	SentDocID	TopicID	SentTopicID	SummarizedSentence
0	22	6	0	23	there are odd ones for the national guard , c...
1	9	10	0	78	the h160 is particularly well suited for long...
2	12	9	0	100	anti submarine warfare asw vsr700 anti submar...
3	12	12	0	103	anti surface warfare asuw vsr700 anti surface...
4	12	16	0	107	maritime security vsr700 maritime security in...
5	12	21	0	112	search and rescue sar vsr700 search and rescu...
6	25	54	1	54	the forum tends to draw a standing room only ...
7	21	6	1	68	data proves customers experience an up to 18 ...
8	21	16	1	78	the fastfin system is approved by the federal...
9	6	17	1	97	h160 timeline 2011 the h160 is first revealed...
10	26	1	2	1	imt vislink s avds is an integrated suite of ...
11	26	20	2	20	including search and rescue operations , narc...
12	20	9	2	30	it includes a 4 axis autopilot , increasing s...
13	3	25	2	63	we used a lot of methodology coming from the ...
14	3	60	2	98	the 169 hils will replace the dauphin , gazel...
15	2	5	3	55	airbus helicopters announced the h160 fuselag...
16	1	21	3	80	ach managing director frederic lemos said , i...
17	13	3	4	28	the aircraft was piloted and monitored from a...
18	10	5	4	48	passing this first step of autonomous flights...
19	24	3	4	60	newly signed up customers include dare county...
20	24	6	4	63	on the support side , airbus signed contracts...

3.3 Méthode abstractive

Comme on a pu voir dans le première partie, le but de la méthode abstractive est de créer des phrases et du texte en se basant sur l'information du texte. C'est un algorithme qui vise à faire un résumé de texte comme un être humain.

Nous avons donc utilisé GPT2, qui permet de générer du texte à partir des morceaux de texte, de phrase ou même des mots.

Afin de tester l'algorithme nous avons, dans un premier temps pris les mots du Topic 0 : "*test aircraft pilot operation base cabin fly customer carry production"* et après une première itération nous avons obtenu le text suivant: *"materials, material items and products in the initial production run of the production facility using a variety of category specific modules. Includes flight services and maintenance; ground*

operations and training; manufacturing, engineering and support; logistics and distribution; transport, logistics and storage; and other duties."

En effet, une itération n'a pas été suffisante pour obtenir un texte correct. Au bout de 22 itérations nous avons obtenu le résultat suivant: *"Cases and complex, multifunctional armaments for various aircrafts around the globe. Whilst carrying out demonstrations of experimental aircraft and measuring the modifications to their appearance and performance, these people form the core of the air forces aviation force and develop flying demonstrations to increase the awareness of the general public".*

Le texte, n'est toujours pas assez correcte et clair.
Dans un second temps, nous avons essayé de mettre en entrée tous les mots de tous les topics et au bout de la troisième itération
GPT2 a créé le texte suivant: *"selection engine design application expo main feed deliver solutions. energy collection electronic engine design software circuit design dynamic flex view development. engine config aerospace circuit testing solution check burn out. test flight aircraft electrical system configuration flight performance module for environmental module test flight flight solid motor test analysis. system liquid fuel delivery 534 body inspection at the pilot's command or by the pilot view descent flow channel flow means at the pilot's command or by the pilot view ascent stream channel flow means at the pilot's command or by the pilot view under the control of a pilot."*

Au bout de la 24ème itération: *" system model aircraft design area date en test job prevent aviation corrections. system air-sea route consider construction soft target. system airplane designer define both care combination test en route correct. system airline pilot passenger interface development soft fly cabin plan. system option fly construction. program automation fly instrument measure alternatives float. program aircraft flight dynamics give airplane options allow diversifications. program program provider soft correct meet delivery. program crew manager post options accelerate credit check destination air carrier propulsion engineering mix field adjustness program aircraft flights en route includes training preparation checks oxygen adjustment soft service".*

Nous pouvons constater que la méthode abstractive n'a pas apporté le résultat attendu. Nous avons donc opté pour utiliser l'algorithme TextRank et donc la méthode extractive. Nous avons donc pu constater ce qu'on a retrouvé dans la littérature : les méthodes extractives donnent de meilleurs résultats que les méthodes de résumé abstractives.

4. Les discussions

- GPT2

L'algorithme n'a pas apporté un bon résumé. En effet, pendant les itérations nous avons pu remarquer que l'algorithme créait des phrases avec des idiomes différents, et un grand nombre de chiffres et de ponctuation.
De plus afin d'avoir un texte correcte on a besoin de beaucoup d'itérations.
Nous avons essayé dans un premier temps de l'implémenter sur Python, mais ce fut un échec. Il serait donc intéressant de trouver une implémentation de GPT2 sur Python pour éviter de passer par une interface.

- Glove

Comme nous avons pu le voir dans nos recherches, Glove est un équivalent de Word2Vec et FastText. Nous voulions le mettre en oeuvre car celui ci est plus efficace que les 2 autres algorithmes. Malheureusement Glove n'est pas encore assez développé pour fonctionner facilement. Le seul environnement sur lequel Glove fonctionne est Google Collab mais cet environnement n'est pas assez sécurisé pour les données que nous avions à traiter.

- Nommer les topics

Enfin, nous avons essayé de nommer les topics. Cela aurait été intéressant pour avoir une idée des mots qui compensent le topic. Dans notre méthode, nous avons décidé d'utiliser les 3 premiers mots de chaque topic pour définir nom du topic. Nous avons réussi à les numéroter mais pas à nommer les topics automatiquement. En tant qu'humain, nous pouvons regarder les mots qui déterminent les topics, mais nous n'avons pas trouvé de code qui nomme les topics automatiquement.

- Méthode de la Distance Cosine

La méthode de la distance cosine expliqué précédemment dans la partie théorique sur les différentes méthodes extractive est une méthode qui a bien fonctionné pour nous sur des petits textes mais que nous n'avons pas pu faire tourner sur les gros corpus d'Airbus car elle demande une trop grande capacité de calcul.
C'est une méthode que Airbus pourrait peut être envisagé d'essayer car elle est simple et produit des résultats assez bon en terme de résumé pure.
Code nécessaire à l'application de cette méthode dans la bibliographie.

5. Conclusion

Durant six mois, nous avons donc oeuvré à trouver une méthode pour résumer au mieux un ensemble de documents d'Airbus Helicopters. Au cours de cette période, il y a eu différentes étapes. Durant les 2 premiers mois nous faisions beaucoup de recherches pour avoir un grand nombre acquérir de la connaissance sur le résumé de document et le NLP.

Nous avions tellement d'idées et de possibilités que nous ne savions pas par quoi commencer. Nous avons donc décidé dans un premier temps de faire des sous groupes pour être efficace. Lorsque nos recherches commençaient à se recentrer sur le sujet, nous voulions continuer à travailler en sous-groupes mais nous nous sommes rendu compte que nous perdions en efficacité.
D'après nos différentes recherches, nos tuteurs pédagogique et professionnel nous ont recommandé de suivre le plan que nous avons déterminé ensemble.

Une fois ce plan établi, nous avons pu travailler ensemble pour avancer étape par étape. Cette méthode de travail fut constructive car nous avons pu sortir 3 méthodes qui résume du texte: deux méthodes extractives (PyTextRank et Tf-Idf) et une méthode abstractive (GPT2) .

Ce Projet de Fin d'Etudes (PFE) clôture donc nos 3 années au Magistère Ingénieur Economiste de l'AMSE. Ce projet avait pour but de nous préparer au monde du travail en travaillant avec des personnes du milieu professionnel.

Pour conclure ce projet, nous souhaitons remercier Flavien Riche (tuteur professionnel) et Pierre Michel (tuteur pédagogique) qui nous ont accompagné et conseillé durant ces six mois.

6. Bibliographie

TF-IDF
Déterminer un score de pertinence (Quentin Fily)
Automatic Extractive Text Summarization using TF-IDF (ASHNA JAIN, april - 2019)

Text Summarization
https://semantive.com/blog/text-summarization-in-python/

https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70

https://towardsdatascience.com/nlp-extracting-the-main-topics-from-your-dataset-using-lda-in-minutes-21486f5aa925

https://towardsdatascience.com/a-quick-introduction-to-text-summarization-in-machine-learning-3d27ccf18a9f

DistanceCosine:
https://towardsdatascience.com/understand-text-summarization-and-create-your-own-summarizer-in-python-b26a9f09fc70

TensorFlow
https://towardsdatascience.com/learn-word2vec-by-implementing-it-in-tensorflow-45641adaf2ac

Word Embedings

https://medium.com/@jessica_51466/les-word-embeddings-la-geométrie-des-mots-4cca8bbc8a2

https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf

https://medium.com/@jonathan_hui/nlp-word-embedding-glove-5e7f523999f6

Transformers
https://talktotransformer.com/

https://openai.com/blog/better-language-models/

https://cai.tools.sap/blog/glove-and-fasttext-two-popular-word-vector-models-in-nlp/

7. Annexes

7.1. Algorithme LDA

Choisir l'ensemble unique de pièces.
Choisir le nombre de composites.
Choisir le nombre de pièces par composite (échantillon d'une distribution de Poisson).
Choisir le nombre de thèmes.
Choisir un nombre non nul et l'appeler alpha.
Choisir un nombre non nul et l'appeler bêta.
Créer le tableau "parties par rapport aux sujets". Pour chaque colonne, tirer un échantillon d'une distribution de Dirichlet (qui est une distribution de distributions) en utilisant le bêta comme entrée. Chaque échantillon remplira chaque colonne du tableau, et donnera la probabilité de chaque partie par sujet (colonne).
Construire le tableau "composites contre sujets". Pour chaque ligne, prélever un échantillon à partir d'une distribution de Dirichlet en utilisant l'alpha comme entrée. Chaque échantillon remplira chaque ligne du tableau, donnera la probabilité de chaque sujet (colonne) par composite.
Construire les composites réels.

Pour chaque composite,

1) rechercher sa ligne dans le tableau "composites-sujets",
2) échantillonner un sujet en fonction des probabilités de la ligne,
3) aller dans le tableau "parties-sujets",
4) rechercher le sujet échantillonné,
5) échantillonner une partie en fonction des probabilités de la colonne,
6) répéter l'étape 2 jusqu'à ce que vous ayez atteint le nombre de parties prévu pour ce composite.

7.2. PAGERANK

PageRank est une métrique introduite en 1995 à l'université de Stanford par Larry Page (un des fondateurs de Google), d'où le nom de "Page" Rank. Cette métrique est utilisée par Google dans son algorithme pour comprendre l'importance d'un site, ou d'une page. L'algorithme évalue la pertinence d'une page donnée et la catégorise dans une échelle de 0 à 10, par le biais de la barre d'outils Google. L'algorithme du PageRank consiste à produire une distribution de probabilité qui sera utilisée pour représenter la probabilité qu'une personne, cliquant au hasard sur des liens, arrive à une page particulière. Le PageRank peut être calculé pour des collections de documents de toute taille. Dans plusieurs documents de recherche, on suppose que la distribution est répartie de manière égale entre tous les documents de la collection au début du processus de calcul. Les calculs de PageRank nécessitent plusieurs passages, appelés "itérations", à travers la collection pour ajuster les valeurs approximatives de PageRank afin qu'elles reflètent plus fidèlement la valeur réelle théorique.

Dans le papier original de Google PageRank est définie de la façon suivante:

$$PR(A) = (1-d) + d\,(PR(T1)/C(T1) + \ldots + PR(Tn)/C(Tn))$$

où nous supposons qu'une *page A* a des pages *T1* à *Tn*. *d* est un facteur d'amortissement qui peut être fixé entre 0 et 1 et qui est généralement fixé à 0,85. *C(A)* est défini comme le nombre de liens sortant de la *page A*.

Le PageRank de n'importe quelle page est approximativement la probabilité que l'internaute aléatoire atterrisse sur une page particulière. Comme plus de liens vont vers les pages importantes, l'internaute a plus de chances de s'y retrouver. Cet algorithme est alors basé sur une chaîne de Markov. En fait, grâce au processus de Markov, le système se déplace d'un état à l'autre, en fonction de la probabilité de l'information qui montre la probabilité de passer de chaque état à tous les autres états possibles.

7.3. TEXTBLOB

TextBlob est une bibliothèque Python pour le traitement de données textuelles. Il fournit une API simple pour plonger dans des tâches courantes de traitement du langage naturel telles que le balisage d'une partie du discours, l'extraction de phrases, l'analyse de sentiments, la classification et la traduction.

Textblob est une librairie très complète qui nous permet de faire de la classification de mots ainsi que de texte et nous permet aussi de corriger l'orthographe et la grammaire dans plusieurs différentes.

Notre but avec Textblob était d'améliorer la qualité de rédaction de notre texte. Notre objectif était de corriger les fautes de grammaire et d'orthographe afin de faire des phrases correctes.

Cependant, nous n'avons pas utilisé cette librairie dans notre code mais elle pourra être utile dans l'avenir du projet lorsque l'on aura des textes dans différentes langues.

7.4. GLOVE

GloVe, Global Vectors for Word Representation, est un algorithme d'apprentissage non supervisé permettant d'obtenir des représentations vectorielles de mots. L'apprentissage est effectué sur des statistiques globales agrégées de co-occurrence mot-mot à partir d'un corpus, et les représentations résultantes présentent des sous-structures linéaires intéressantes de l'espace vectoriel des mots.

GloVe est essentiellement un modèle log-bilinéaire avec un objectif de moindres carrés pondérés. La principale intuition qui sous-tend le modèle est que les rapports de probabilités de cooccurrence mot à mot ont le potentiel d'encoder une certaine forme de signifiance.

Le but de GloVe est donc d'apprendre les vecteurs de mots de telle sorte que leur produit scalaire soit égal au logarithme de la probabilité de co-occurrence des mots. Comme le logarithme d'un rapport est égal à la différence des logarithmes, cet objectif associe les rapports des probabilités de co-occurrence aux différences vectorielles dans l'espace des vecteurs de mots. Il crée les vecteurs de mots qui fonctionnent bien pour les tâches d'analogie de mots et les tâches de similarité et de reconnaissance d'entités nommées. GloVe se différencie de Word2vec qui apprend par des phrases de façon continue. Il prédit les mots voisins en maximisant la probabilité qu'un mot se produise étant donné un mot central en effectuant une régression logistique dynamique.

GLoVe c'est une autre méthode de word embedding mais qui utilise un mécanisme et une équation différentes dans la création de la matrice d'embedding. Pour mieux comprendre cet algorithm on a besoin de définir les termes suivants:

X_{ij} tabulate the number of times word j occurs in the context of word i.

$X_i = \sum_k X_{ik}$

$P_{ij} = P(j|i) = X_{ij}/X_i$

Le ratio de de probabilité de co-occurrence est défini de la façon suivante:

$w \in \mathbb{R}^d$ are word vectors — probe word

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

co-relations between the word w*i* and w*j* — co-occurrence probabilities for the word w*j* and w*k*

Ce ratio va nous permettre de voir la corrélation du mot sonde wk avec le mot w_i et w□.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Very small or large:
solid is related to ice but not steam, or
gas is related to steam but not ice

close to 1:
water is highly related to ice and steam, or
fashion is not related to ice or steam.

Ce rapport peut peut être petit, grand ou égal à 1 selon leurs corrélations. Ce rapport nous permet donc de voir les relations entre trois mots différents. Par exemple, si le rapport est grand, le mot sonde est lié à w_i mais pas à w□. On pourrait comparer cela à un un bi-gramme et/ou un 3-gramme.

Le but est de développer un modèle pour F en tenant compte de certains comportements souhaitables pour le vecteur d'incorporation w. Pour cela on peut calculer la différence et la similarité de l'incorporation des mots pour les paramètres de F:

$w_i^T \tilde{w}_k$ relate to (high probability if they are similar)

$$F\left((w_i - w_j)^T \tilde{w}_k\right) = \frac{P_{ik}}{P_{jk}}$$

$w_j^T \tilde{w}_k$

Il est import de garder la relation linéaire entre tous ces embedding vectors. Pour cette raison on rend F(x) une fonction exponentielle avec F(x) = exp(x). En remplaçant on obtient:

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(X_{ik}) - \log(X_i)$$
$$w_i^T \tilde{w}_k + b_i + \tilde{b}_k = \log(X_{ik})$$

co-occurrence count for word w*i* and w*k*

Afin de maximiser la probabilité qu'un mot se produise nous avons besoin de la fonction de coût et d'utiliser l'erreur quadratique moyenne pour déterminer l'erreur. En effet, les paires de mots ont des fréquences d'occurrence différentes dans le corpus, et donc pour prendre en compte cela nous avons besoin de définir un poids qui permettra de réajuster le coût de chaque paire de mots. Ce poids est défini par la fonction *f* suivante:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

(100 : $x_{\max}$, 3/4 : α)

Si le nombre de cooccurrences atteint ou dépasse un certain seuil, le poids sera de 1. Ce poids dépend du nombre de cooccurrences.

7.5. GRIDSEARCH

La plupart des modèles de machine learning doivent être paramétrés pour donner les meilleurs résultats. Par exemple pour un Random Forest, on doit choisir le nombre d'arbres à créer et le nombre de variables à utiliser à chaque division d'un noeud. Si on paramètre à la main, cela peut vite s'avérer très coûteux en temps (et pas forcément très intéressant) …

C'est là que le Grid search intervient. C'est une méthode d'optimisation (hyperparameter optimization) qui va nous permettre de tester une série de paramètres et de comparer les performances pour en déduire le meilleur paramétrage.

Il existe plusieurs manières de tester les paramètres d'un modèle et le Grid Search est une des méthodes les plus simples. Pour chaque paramètre, on détermine un ensemble de valeurs que l'on souhaite tester. Par exemple, dans le cas d'un Random Forest on pourrait tester :

- Nombre d'arbres : {100, 150, 200, 250}
- Nombre de variables sélectionnées : {8, 10, 12, 14, 16}
-

Le Grid Search croise simplement chacune de ces hypothèses et va créer un modèle pour chaque combinaison de paramètres. Dans notre exemple nous aurons

20 modèles à construire. Vous comprenez donc rapidement qu'il ne faut pas abuser du Grid Search parce qu'il augmente considérablement les temps de calcul.

Jusque là rien de bien compliqué, on a juste créé une grille de paramètres. Voyons maintenant comment tester intelligemment ces 20 modèles sur le même dataset. Nous allons utiliser une méthode de validation croisée : le k-fold.

La méthode consiste à découper le data set en k échantillons. On sélectionne x échantillons pour constituer l'échantillon d'apprentissage. Les k-x échantillons restants permettront d'évaluer la performance du modèle. Pour construire le modèle suivant on sélectionne les échantillons différemment de manière à ne jamais avoir les mêmes échantillons d'apprentissage et de validation.
Une fois que chaque modèle a pu être entrainé et évalué, il ne reste plus qu'à comparer la performance pour choisir le meilleur modèle.
Dans notre cas par exemple, c'est le modèle utilisant 150 arbres et 10 variables sélectionnées à chaque division d'un noeud qui est le meilleur. La méthode Grid Search nous permet donc d'optimiser notre modèle.
Grid Search a aussi ses limites puisque c'est vous qui définissez à l'avance les paramètres que vous voulez tester.