



Claims and Tweets : mise en correspondence de ClaimsKG et TweetsCov19

22 Mai 2022

Aiouaz Fatimah-Zahra
El-Haddouchi Nour Eddine
Paygambar Karl

Sommaire

1	Introduction	3
2	Présentation des données	4
2.1	ClaimsKG	4
2.2	TweetsCov19	5
2.3	Claims2	5
3	Association de phrases	6
3.1	Analyse Textuelle	6
3.2	Analyse Vectorielle	12
3.3	Comparaison des méthodes	18
4	Regroupement par sujets	19
4.1	Regroupement par K-Moyennes	20
4.2	Allocation Latente de Dirichlet	21
4.3	Vers l'application	24
5	Modules complémentaires	25
5.1	Création d'une base Claims-TWeets	25
5.2	Filtrage des textes	27
6	Gestion de Projet	28
7	Conclusion	29
Références		30
A	Annexes	32

1 Introduction

L'accès démocratisé à la communication en réseau entraîne un afflux de données conséquent, non sans effets sur la société et ses individus. Pendant les récentes crises sanitaires, humanitaires et économiques, les réseaux sociaux ont été le théâtre de diverses campagnes de propagande, qui se retrouvent relayées en chaîne. Des organismes s'attachent donc à faire de la vérification d'information afin de distinguer les affirmations sans fondement, d'informations vérifiées.

Des bases de données référençant les affirmations, tel que ClaimsKG¹, permettent de faire de l'exploration de données dans le but de rendre ces vérifications automatiques. L'ensemble des interactions sur les plateformes d'échange, comme Twitter, est aussi une source qui va permettre d'étudier les comportements. TweetsKB, et sa partie subséquente TweetsCov19², permettent quant à elle d'exploiter les commentaires pour essayer de faire sortir des modèles.

Le projet de recherche consiste ici en la création de lien entre ces deux types d'éléments. En considérant un utilisateur final, il devra être possible de pouvoir trouver des points communs entre des assertions vérifiées, et un ensemble de commentaire. Cela permettrait de voir comment évolue la communauté autour d'une assertion, voir même comment cette assertion a été créée. Les bases de données fournissent en plus des textes des ensembles de métadonnées, telles la date de parution ou l'auteur, permettant de faire un lien potentiel entre des événements et le texte en question. Relier claims et tweet pourrait se faire autour de cet événement, en cherchant tout les éléments se rapportant au même sujet, ou alors un tweet en particulier d'apparence isolé pourrait se voir lié à une affirmation, présente ou passée. L'important ici est de créer une correspondance entre une base de donnée comportant des éléments vérifiés et tracés avec d'autres données "réelles", ce qui aurait de nombreuses applications allant de l'historique d'affirmation à des études comportementales à grande échelle. C'est dans ce cadre que deux principales tâches seront abordées ici.

Dans un premier temps, il s'agit de mettre en correspondance des éléments deux-à-deux. Pour un tweet donné, il peut être intéressant de trouver l'assertion la plus similaire, dans le but de retrouver la source du commentaire. Au contraire une assertion peut trouver son origine dans un tweet, comme ça a souvent été le cas pour les sujets autour de la politique notamment. Pour celà, deux approches ont été étudiées. Les méthodes basées sur l'analyse syntaxique et sémantique vont être testées pour trouver une métrique permettant de satisfaire cette tâche. Ensuite, des techniques basées autour de la transformation en vecteur de valeurs numériques seront elles aussi étudiées pour la même tâche. Une comparaison sera faite entre les résultats de ces deux approches.

La seconde tâche sera de faire une analyse par sujet des éléments. Pour ne pas se satisfaire que d'une réponse, il est possible de ranger par groupe les tweets et claims. Celà peut mettre en évidence les échanges autour d'un claim, mais aussi de grouper des éléments qui feraient en fait partie d'une même vague de désinformation, sur plusieurs années par exemple. Dans ce but, le KMeans sera comparé à l'allocation latente de Dirichlet, et des perspectives seront étudiées pour les deux méthodes.

Avant de conclure, certaines des méthodes vues au cours du déroulement du projet seront utilisées afin de proposer une méthode pour créer une base de données de référence de Claims et Tweets, dont la correspondance a été annotée. De plus, les méthodes de filtrage faites à partir de ce qui a été utilisé notamment pour l'analyse sémantique seront proposées, dans le but de former un outil de recherche.

1. <https://data.gesis.org/claimskg/>

2. <https://data.gesis.org/tweetscov19/>

2 Présentation des données

2.1 ClaimsKG

ClaimsKG est une base de données structurée qui fait office de registre des allégations, vérifiées par des organismes dédiés [Tchechmedjiev et al. \[2019\]](#). La base de données est construite autour d'un graphe (figure 12) de connaissances contenant les allegations, mais aussi des métadonnées telles l'auteur, ainsi que l'idice de véracité servant ici de référence.

Les Caractéristiques extraites et fournies sont :

- **text** : le texte de la déclaration
- **truthRating , ratingName** : sa valeur de vérité ou sa valeur nominale, à la fois une valeur nominale normalisée et la valeur originale
- **link** : un lien vers l'évaluation de l'affirmation depuis le site de vérification des faits
- **author** : l'auteur de l'affirmation et l'auteur de la critique de l'affirmation
- **keywords** : un ensemble de mots-clés extraits des sites web de vérification des faits qui agissent comme des topics
- **headline** : le titre de l'article de révision
- **date** : la date de publication de la déclaration et la date de publication de l'article de synthèse
- **named-entities-claim , named-entities-article** : les entités nommées extraites du corps de la déclaration et du corps de l'article de révision.

Le système d'évaluation est composé de quatre catégories de base (TRUE, FALSE, MIXTURE, OTHER), a été retenu. Les affirmation catégorisées strictement sont les TRUE/FALSE, et la catégorie MIXTURE représente un élément qui se situe quelque part sur une échelle de vérité. Lorsque ce n'est pas le cas, le choix de l'option OTHER s'impose.

TABLE 1 – Statistiques concernant les métadonnées des affirmations (en mai 2022)

Property	Global	Snopes	Politifact	Africa Check	Truth or Fiction	Check your Fact	FactsCan	Fullfact	AFP Factcheck	AFP Factuel (FR)
Number of claims	33,261	12,812	16,476	520	1,311	823	125	250	678	266
Claim text	99.10%	99.98%	100.00%	100.00%	100.00%	100.00%	100.00%	0.00%	100.00%	100.00%
Claim author	44.57%	0.00%	100.00%	0.00%	0.00%	0.00%	100.00%	39.54%	82.32%	62.40%
Claim date published	44.74%	0.00%	99.91%	0.00%	0.00%	0.00%	98.40%	0.00%	100.00%	100.00%
Claim with citation	81.00%	82.32%	76.95%	96.97%	100.00%	99.76%	100.00%	0.00%	99.55%	95.48%
Claim keywords	77.10%	67.47%	100.00%	99.88%	0.00%	0.00%	100.00%	100.00%	0.00%	0.00%
Claim entity mention	98.82%	99.66%	99.98%	98.56%	100.00%	100.00%	100.00%	0.00%	100.00%	100.00%
Claim review URL	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Claim review headline	99.11%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	0.00%	100.00%	100.00%
Claim review author	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Claim review date published	77.78%	100.00%	0.00%	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Claim review language	100.00%	100.00%	0.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Claim review entity mention	62.95%	67.98%	55.83%	72.49%	73.32%	64.92%	96.00%	43.55%	74.10%	53.13%
Claim review language	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Identical claims	87	38	49	0	0	0	0	0	0	0
True claims	4,404	1,846	2,334	60	125	0	39	0	0	0
False claims	12,350	6,809	5,036	211	246	0	48	0	0	0
Mixture claims	10,834	1,925	8,844	0	63	2	0	0	0	0
Other claims	5,706	2,232	262	282	877	821	38	250	678	266

Le modèle de données pour ClaimsKG est basé sur des termes provenant de terminologies établies par schema.org, NLP Interchange Format (NIF), et Internationalization Tag Set (ITS). La figure 13 illustre un exemple de claimReview par Politifact pour un claim prononcé le 6 juin 2018 par Donald Trump. Nous pouvons constater qu'il existe deux instances de schema :Rating, l'une pour la notation originale de Politifact et l'autre pour la notation normalisée fournie par la base ClaimsKG. Au-delà des informations sur les métadonnées, nous pouvons également constater que le review mentionne le nom d'entité "White House", qui correspond probablement à la résidence du président des Etats-Unis.

2.2 TweetsCov19

TweetsCOV19 [Dimitrov et al. \[2020\]](#) - un sous-ensemble de TweetKB contenant des tweets en rapport avec la pandémie de COVID-19 de manière pré-traitée, et permet d'effectuer différentes tâches (Sentiment Analysis, Topic Analysis...) sur des données textuelles spécifiquement en rapport avec le Covid.

Le jeu de données TweetsCOV19 comprend 8 151 524 tweets originaux postés par 3 664 518 utilisateurs capturés entre octobre 2019 et avril 2020.

TABLE 2 – Statistiques descriptives de TweetsCOV19

Feature	Total	Unique	Ratio of tweets with at least one feature
Hashtags :	3,653,928	566,308	0.30
Mentions :	5,363,449	1,251,963	0.40
Entities :	11,537,537	331,307	0.70
Non-neutral sentiment :	4,478,603	-	0.55

La majorité (70%) des tweets contiennent des entités, ce qui pourrait servir pour certains tests.

TABLE 3 – Les cinq premiers mots clés pertinents, les mentions d'utilisateurs, les hashtags et les domaines payants des TweetsCOV19

keywords	frequency	user mentions	frequency	hashtags	frequency	PLDs	frequency
ppe	3,368,192	realdonaldtrump	41,839	covid19	160,585	twitter.com	251,839
coronavirus	2,363,080	narendramodi	13,039	coronavirus	148,317	youtube.com	99,505
covid	2,308,054	pmoindia	12,701	covid_19	27,049	instagram.com	50,846
corona	1,513,195	jaketapper	9,836	stayhome	26,542	nytimes.com	30,892
covid19	1,498,386	who	9,776	china	23,602	theguardian.com	26,737

Le tableau 3 illustre les cinq principaux mots-clés, mentions d'utilisateurs, hashtags et PLD les plus fréquemment trouvés dans TweetsCOV19. Le mot-clé qui apparaît le plus souvent -ppe- est l'acronyme de (personal protective equipment) équipements de protection individuelle tels que les masques, les lunettes de protection, et les gants. Les hommes politiques, les médias et les organisations de santé sont les mentions d'utilisateurs les plus récurrentes, tandis que @realdonaldtrump est de loin l'utilisateur le plus souvent mentionné . À priori que plusieurs sujets seront mentionnés, et possiblement dans un seul et même tweet. Cela peut être intéressant durant la phase de regroupement par sujet. Les deux schémas en annexe 12 et 13 illustrent le model RDF de la base TweetsCOV19 ainsi qu'un exemple de version instanciée.

2.3 Claims2

La base de données Claims-Claims, ou Claims2, servira de référence pour la plupart des méthodes qui seront testées dans ce projet. Ces données, mises à disposition par l'encadrant dans le projet, sont issues de ClaimsKG. Il s'agit de 584 claims couplés par paires, et annotés en fonction de leur proximité. Elle dispose des mêmes attributs que ClaimsKG, et notamment les textes complets des claims. Celà permet d'avoir une donnée "brute", non pré-traitée, et d'avoir plus de liberté sur les traitement applicables ainsi que sur les méthodes qui pourront être utilisées. De plus, les annotations vont à la fois caractériser des éléments qui sont exactement similaires, que d'autes qui appartiennent aux mêmes sujet (Annexe 14). Il sera donc possible d'utiliser ces références pour plusieurs types de tâches.

3 Association de phrases

La première tâche du projet est de faire des associations directes entre les claims et les tweets. Plus généralement, les méthodes utilisées vont faire associer des textes entre eux, dans le but d'être ensuite appliquées à des données réelles. Deux types d'approches sont envisageables dans ce cas.

Afin de déterminer la proximité entre deux chaînes de caractères, des analyses sémantiques ou syntaxiques de leur composition pourraient permettre de créer une mesure attestant de la similarité des deux textes. De manière analogue, des méthodes plus avancées basées sur la sémantique, en cherchant le "sens" des phrases, pourraient aussi être utilisées pour réaliser cette même tâche. L'ensemble de ces méthodes sera vu en partie **3.1**.

D'un autre côté, les algorithmes de classification utilisent traditionnellement des méthodes permettant de transformer les chaînes de caractère en vecteurs, basés sur l'occurrence de mot. Il est ainsi possible de déterminer des mesures de proximité en se basant sur ces vecteurs, et les approches vectorielles seront vues en partie **3.2**.

Dans les deux cas, il s'agira de se rapprocher des annotations produites sur claims2, notamment en faisant évoluer des seuils de similarité pour différentes mesures. Le but est de trouver une tendance, et une association entre un seuil et une méthode qui pourrait s'appliquer à un lien Claims-Tweet.

3.1 Analyse Textuelle

Pour mesurer la similarité textuelle entre deux textes, deux approches peuvent être utilisées : l'approche syntaxique et l'approche sémantique. Étant donné que l'utilisation de cette dernière nécessite un travail considérable pour sa mise en place, nous avons privilégié une approche syntaxique.

3.1.1 Approche syntaxique : Fonctions Levenshtein, Jaccard, LCS

La similarité syntaxique permet de comparer des documents textuels sous forme de chaînes de caractères, en fournissant une valeur numérique permettant de faire une comparaison. Nous pouvons ainsi citer plusieurs mesures tels que Levenshtein, Jaccard ou encore LCS que nous étudierons à travers différentes comparaisons et pour l'écriture de différents programmes.

D'abord, la fonction de Levenshtein permet de calculer la distance et le ratio. La distance calcule le nombre minimal des opérations de suppression, d'insertion ou encore de substitution pour transformer une chaîne de caractère en une autre chaîne. Elle est obtenue par le coût de chaque opération et est égal à 1 sauf dans le cas d'une substitution de caractère identique. Ainsi plus la distance est proche de 0 plus la correspondance est élevée. Le ratio effectue le même calcul mais ne donne pas la même importance et donc le même coût à la suppression et à l'insertion.

Ensuite, l'indice de Jaccard, quant à lui permet de calculer le rapport entre la cardinalité (la taille) de l'intersection des ensembles considérés et la cardinalité de l'union des ensembles." [Negre \[2013a\]](#). Le résultat est compris entre 0 et 1 avec 0 pour aucune correspondance.

Enfin, la fonction LCS détermine la plus grande sous-séquence commune entre deux textes. Pour une correspondance totale, le résultat est égal à la longueur de la chaîne.

3.1.2 Comparaison sur le texte brut.

Premièrement, nous avons appliquées ces différentes fonctions sur le texte brut pour pouvoir avoir une première approche sur les résultats.

Levenshtein	E		E*		ST		N	
Texte A vs Texte B	1	2	4	11	5	6	255	258
Ratio	1.0	1.0	0.53	0.48	0.54	0.61	0.51	0.40
Distance	0	0	55	59	63	48	89	83

On constate que pour la similarité de deux textes avec une correspondance totale (annoté ici par un E), on a par exemple testé le texte A.1 avec le texte B.1 on a comme résultat un ratio de 1.0 et une distance de 0. Toutefois, pour les autres références on ne constate pas réellement de différence dans les chiffres et on ne peut donc pas déterminer facilement quel texte est annoté E*, ST ou encore N.

Deuxièmement, pour pouvoir comparer les résultats nous avons appliquées les mêmes données sur l'indice de Jaccard.

Jaccard	E		E*		ST		N	
Texte A vs Texte B	1	2	4	11	5	6	255	258
Indice de Jaccard	1.0	1.0	0.63	0.58	0.75	0.8	0.71	0.59

Ces résultats ne nous permettent pas encore de dissocier E*, ST et N mais nous pouvons distinguer E donnant un résultat de 1.0. Enfin, nous les avons appliquées sur la fonction LCS.

LCS	E		E*		ST		N	
Texte A vs Texte B	1	2	4	11	5	6	255	258
La plus grande sous séquence commune	100	88	46	40	50	44	59	43

L'utilisation de ces trois fonctions sur les données brutes n'a pas été concluante aux premiers abords, sauf pour les cas extrêmes tels que la similarité totale ou la non-similarité. Pour avoir de meilleurs résultats nous avons décidé d'avoir une approche qui permet de traiter le texte en amont.

3.1.3 Nettoyage du texte

D'abord, l'écriture d'une première fonction qui permet de supprimer les mots "inutiles" tels que les déterminants, les pronoms, les conjonctions etc... pour ne laisser que le sujet, le verbe et les compléments. On passe ainsi par exemple pour le texte "Hillary Clinton says she helped usher Iran to the negotiating table" à " Hillary Clinton says helped usher Iran negotiating table".

Ensuite, l'écriture d'une seconde fonction permettant de conserver les Noms Propres d'une phrase pour permettre une comparaison des sujets, on aurait par exemple pour cette phrase " U. S. Rep. David Cicilline says that when he was mayor of Providence, the city's internal auditor was not blocked from accessing financial records " le résultat suivant "U. S. Rep. David Cicilline Providence".

3.1.4 Ecriture programme

A - A partir d'une partie des résultats données

Pour pouvoir écrire quelques programmes, nous allons effectuer le calcul pour une partie des données et ainsi comparer les résultats en fonctions des annotations. Nous avons étudié 9 textes pris aléatoirement, et avons appliquées la distance et le ratio de Levenshtein sur le texte modifié, la distance et le ratio des Noms Propres, et enfin la plus longue sous séquences communes.

	ST			E*			N		
Ratio	32	81	53	54	48	75	54	52	45
Distance	54	15	48	55	59	19	69	69	62
Ratio Noms Propres	39	85	69	58	36	63	75	51	74
Distance Noms Propres	40	5	14	18	26	12	12	52	16
Jaccard	56	87	68	63	58	66	77	88	60
LCS	19	50	39	46	40	40	59	54	49

En analysant les résultats on a réussi à déterminer quelques propriétés :

- Si la Distance est plus grande que 60 on a forcément un texte ayant pour annotation N.
- Si LCS est plus petite que 20 on a comme résultat un texte égal à ST.
- Si le Ratio Noms Propres est plus grand que le Ratio on a un texte pour annotation égal à ST.
- Si LCS est compris entre 40 et 50 c'est une correspondance à E* ou N.
- Si la Distance est plus petite que 60 on a une correspondance de E* ou ST.

À partir de ces éléments, nous pouvons écrire un programme généralisé sur toutes les données. Dans une hypothèse où les calculs correspondent à plusieurs critères, on aurait une liste de possibilité avec les annotations pouvant contenir l'annotation réelle. Par exemple, si après comparaison du texte la distance est plus petite que 60 et que LCS est compris entre 40 et 50 le résultat correspond à [E*,N,ST] et nous pouvons ainsi comparer le résultat avec la réelle annotation.

Pour pouvoir vérifier l'exactitude des résultats du programme, on écrit une fonction qui calcule le pourcentage de réussite. Ici le pourcentage de réussite est volontairement maximisé en utilisant une liste de possibilités. Malgré cela, les résultats ne sont pas concluants puisque l'on a un pourcentage de réussite de 73%.

On continue par l'écriture d'un second programme un peu plus précis, qui comptabilise un système de point. Ce programme aurait pour objectif qu'à partir des seuils donnés, on calcule le score potentiel de chaque résultat. Si un résultat a plus de chance de se réaliser, il est sélectionné. Pour reprendre l'exemple ci-dessus, la liste [E*,N,ST,E*] attribue plus de points à E* qu'à N et ST. Ce programme a permis de déterminer un pourcentage de réussite de 76%.

B - À partir de la moyenne

Après avoir déterminé des seuils à partir de quelques données, nous allons déterminer des seuils à partir de la moyenne de toutes les données. Nous avons pu remarquer :

- Si la distance est plus petite à 40 c'est égal à l'annotation E
- Si LCS est plus grand que 55 c'est égal à l'annotation E

- Si la Distance est comprise entre 40 et 60 est égal à E* et ST
- Si la Distance est plus grande que 65 l'annotation est égal à N
- Si le ratio est compris entre 0.5 et 0.6, l'annotation correspond à E* et ST
- Si le ratio est plus petit que 0.40 on a N.

Le programme écrit après l'analyse de ces données n'aura cependant qu'un taux de réussite de 54%.

C – À partir d'un algorithme automatisé

Après avoir calculé les seuils à partir d'un échantillon de données, on décide de créer un programme qui calcule automatiquement les seuils avec le meilleur résultat. On lance ces calculs pour la distance et le ratio de Levenshtein.

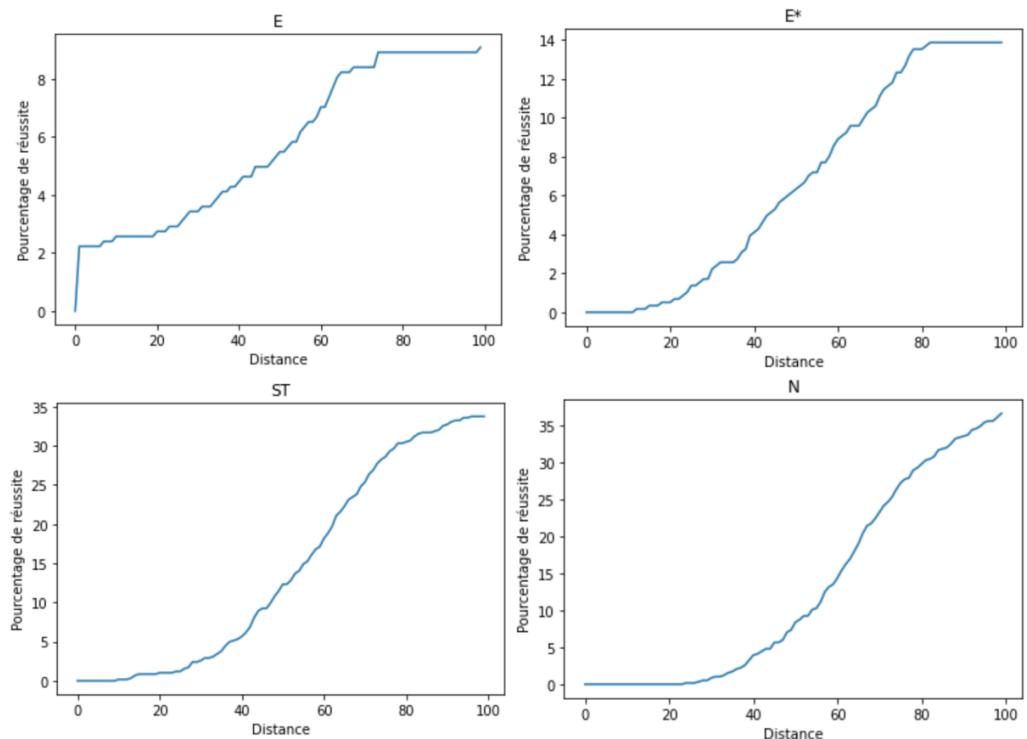


Figure 5 – Distance de Levenshtein en fonction du seuil

On constate ici que plus la distance est élevée plus le pourcentage de réussite est élevé. Les taux de réussite sont bas mais, nous constatons que la courbe de N et ST est sujette à une plus grande interprétation et que la distance seule ne permet pas de distinguer ces deux annotations.

Nous avons pu effectuer les mêmes graphiques pour le ratio.

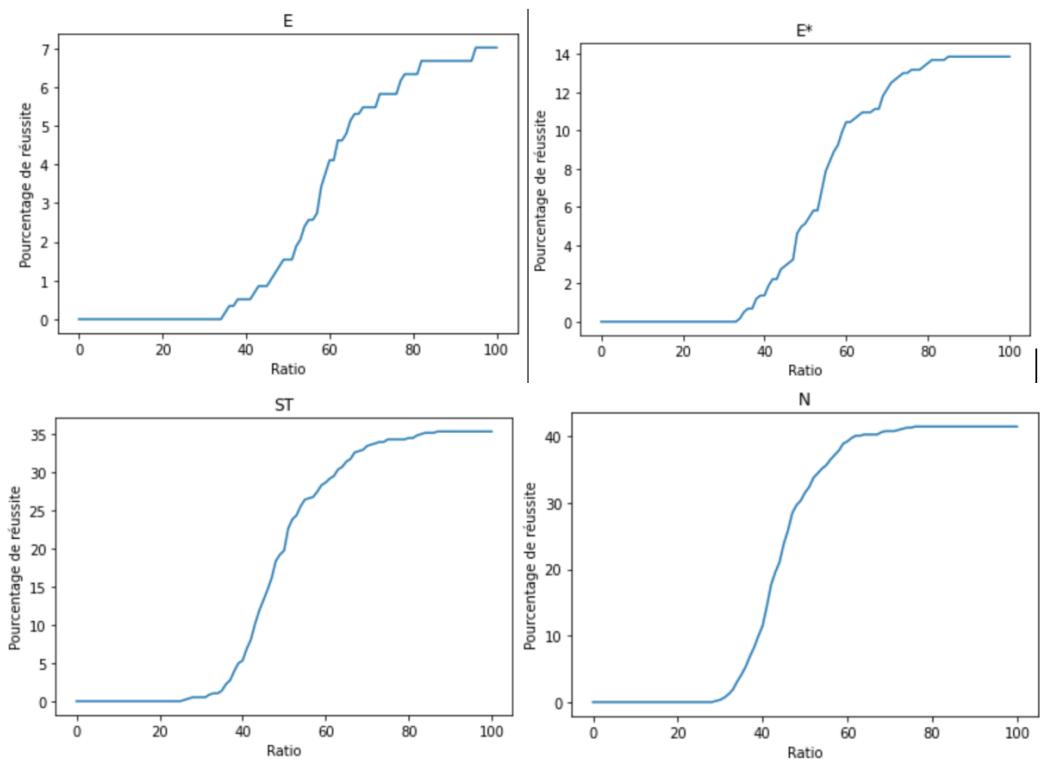


Figure 6 – Ratio de Levenshtein en fonction du seuil

Nous avons pu conclure qu'une distance plus élevée et un ratio plus élevé, avaient une plus forte probabilité de correspondre à l'annotation N ou ST. Cependant, pour les autres annotations, le taux de réussite est très bas.

3.1.5 Études des résultats : Matrices confusion, Rappel, Précision, F-score

A partir de tout ces calculs, on écrit un programme final qui renvoie une annotation. On peut ainsi comparer l'annotation donnée avec la vraie annotation entre le texte A et le texte B. Dans le cas où le résultat ne se situe dans aucun seuil, on a décidé de le noter "ST" pour pouvoir calculer la matrice de confusion. Afin de faciliter l'interprétation de cette matrice, nous avons calculé la précision, le rappel et le f-score.

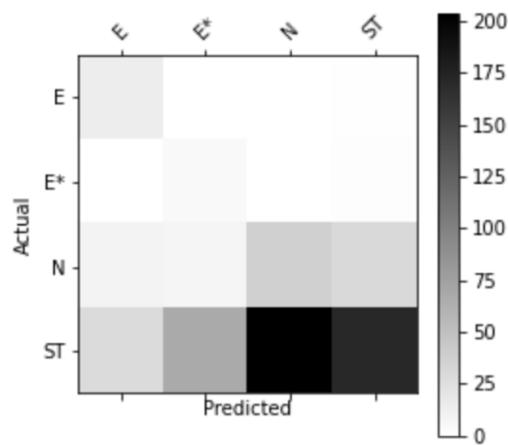
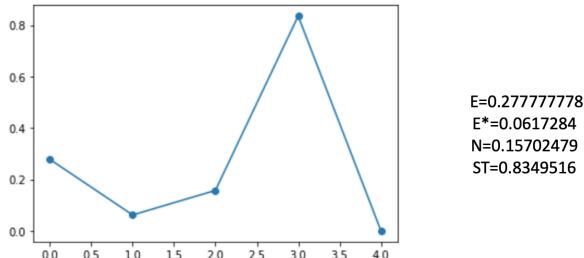
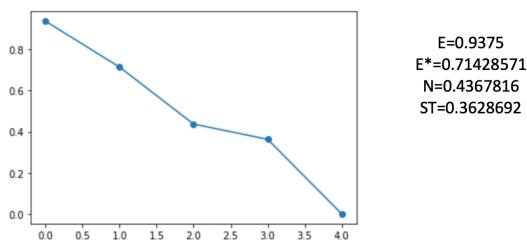


Figure 7 – Matrice de confusion

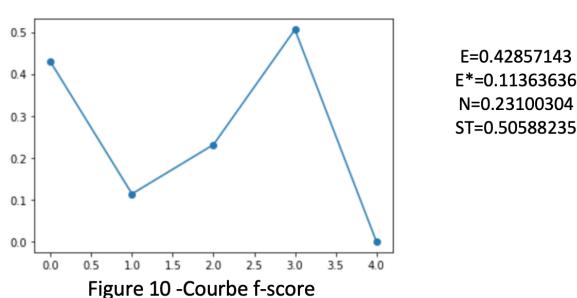
On constate à partir de cette matrice de confusion que le programme final a majoritairement donné l'annotation ST, ce qui signifie qu'il a certainement déterminé cette annotation à partir d'un seuil soit par défaut. Le seuil semble beaucoup trop bas, et les éléments ont tendance à être classés comme non correspondants, ou à sujets similaires. Ces résultats ne sont pas concluants.



On constate à partir de la précision que le nombre de prédictions positifs bien effectuées par le programme est le plus élevée pour l'annotation ST puis bien après par l'annotation E. On peut expliquer ce résultat en raison d'un choix par défaut pour l'annotation ST et par la forte probabilité de trouver l'annotation E au cours du programme.



Dans ce cas, le rappel est le plus élevé pour l'annotation E.



Enfin le f-score maximum correspondant à l'annotation ST est très faible (0.50). Ainsi, le programme n'est pas très pertinent car offre un résultat mauvais malgré le biais sur la valeur ST.

3.1.6 Application sur les données ClaimsKG et Tweets

À partir des données de références, nous pourrons à présent nous consacrer à une analyse des tweets et des claims.

A- Application des travaux

On applique sur le premier tweet, Levenshtein et Jaccard pour pouvoir le comparer avec tous les autres claims en calculant leurs similarités. Le premier tweet étant «Trump needs to immediately divest from his businesses and comply with the emoluments clause. Iran could threaten Trump hotels *worldwide* and he could provoke war over the loss of revenue from skittish guests. His business interests should not be driving military decisions. — Ilhan Omar (@IlhanMN) January 6, 2020»

On a nettoyé en amont le texte en supprimant tous les mots dits « inutiles » précédemment écrits pour les données de références. On se retrouve donc avec « Trump needs immediately divest from businesses comply emoluments clause. Iran could threaten Trump hotels *worldwide* could provoke war over loss revenue from skittish guests. His business interests should be driving military decisions. — Ilhan Omar (@IlhanMN) January 6, 2020 »

Nous avons appliqué la méthode Levenshtein, ce qui a donné comme 5 meilleures résultats les claims 302,520,553,398 et 645. Le claim ayant la distance la plus basse étant le claim 302 avec une distance de 205 contre 207 et 209 pour les autres. Le claim 302 correspond à « A commencement speech delivered by President Trump at Liberty University in May 2017 bore remarkable similarities to one given by Reese Witherspoon's character in the 2001 film "Legally Blonde." ». Comme nous pouvons le constater le claim et le tweet parle tout deux du président Trump mais ils ne parlent pas du même sujet.

Ensuite, nous appliquons Jaccard qui a donné comme 5 meilleures résultats, les claims 215,608,622,668 et 889. Après analyse des claims, on constate que le claim 215 correspond parfaitement au tweet 1, sachant que le claim 215 correspond à « In January 2020, U.S. Rep. Ilhan Omar advised Iran to attack Trump-branded hotels in the world, thus committing treason. »

B - Application du programme

Nous pouvons appliquer le programme ayant eu le taux de réussite le plus élevé pour les références sur ces nouvelles données, afin d'offrir un aperçu des résultats. Celui-ci a déterminé une seule correspondance pour le premier tweet qui correspond selon lui au claim 974. Le claim 974 étant «Text reproduces a», le retour du programme n'a aucun lien avec cette valeur.

Au final, l'approche syntaxique nous a permis d'évaluer la similarité entre deux chaînes de caractères et ainsi faire correspondre au mieux les données de claims et de tweets. Toutefois, elle ne prend pas en compte la sémantique, et deux termes ayant le même sens ne sont pas repérés comme tel. Au contraire, un terme polysémique va être pris en compte dans le résultat alors que celui-ci désignera un sens différent. Cette approche ne prend pas non plus en considération les relations syntaxiques. Autrement dit, la fonction des mots est ignorée alors que celle-ci joue un rôle important dans la détermination du sens de la phrase.

3.2 Analyse Vectorielle

Le traitement du texte en langage naturel associé aux techniques d'apprentissage automatique nécessite de convertir la chaîne de caractères en un vecteur de réels. Les documents sont représentés comme des vecteurs de caractéristiques, qui représentent les termes qui figurent dans la collection. La valeur de chaque caractéristique est appelée le poids du terme, et est généralement une fonction de la fréquence du terme dans le document.

Ainsi, en prenant la fréquence de chaque terme comme poids, les mots qui apparaissent les plus souvent sont considérés comme plus importants et donc représentatifs du document [Negre \[2013b\]](#). La représentation d'un document sous forme de vecteur s'effectue en 2 étapes :

1. extraire les termes pertinents du document : le prétraitement du contenu des documents textuels en supprimant les mots vides, la ponctuation et éventuellement les "retours chariot", puis en procédant à la lemmatisation et à la segmentation du texte.
2. calculer les poids des termes restants : On peut obtenir le poids de chaque terme dans un document

de différentes manières : booléen, fréquence des termes, tf-idf (Term frequency - Inverse Document Frequency).

3.2.1 Approche "TF-IDF"

Le TF-IDF permet de mesurer l'importance d'un terme présent dans un document, vis-à-vis d'une collection. Son poids croît de manière proportionnelle au nombre d'occurrences du mot dans le document, mais est aussi fonction de la fréquence du mot dans la collection. [Tata and Patel \[2007\]](#). La TF-IDF a pour but de donner un poids plus important aux termes les moins fréquents, considérés comme les plus discriminants.

De nombreuses formes existent, mais en général elles fournissent un poids pour chaque terme ayant une valeur normalisée entre 0 et 1 ; une valeur proche de 0 indique un terme de faible pertinence, tandis qu'une valeur proche de 1 indique une pertinence élevée. Les mots qui sont fréquemment utilisés dans un seul ou un petit groupe de documents sont associés à des valeurs plus élevées que les mots communs tels que les articles et les prépositions.

Étant donné une collection de documents D, un mot m, et un document individuel d ∈ D, la valeur de la TFIDF ici notée m_d est :

$$m_d = f_{m,d} * \log\left(\frac{|D|}{f_{m,D}}\right) \quad (1)$$

où $f_{m,d}$ est égal au nombre de fois où m apparaît dans d, $|D|$ est la taille du corpus, et $f_{m,D}$ est égal au nombre de documents dans lesquels m apparaît dans D.

Il y a quelques situations différentes qui peuvent se produire ici pour chaque mot, en fonction des valeurs de $f_{m,d}$, $|D|$, $f_{m,D}$.

Admettons que $|D| \propto f_{m,D}$, c'est-à-dire que la taille du corpus est approximativement égale à la fréquence de m sur D. Si $1 < \log(|D| / f_{m,D}) < cst$ pour une très petite constante cst, alors m_d va être plus petit que $f_{m,d}$ mais toujours positif. Cela implique que m est relativement commun dans l'ensemble du corpus mais conserve néanmoins une certaine importance dans D. Par exemple, cela pourrait être le cas si le TF-IDF examinait le mot "Corona" dans le corpus des documents de l'Organisation mondiale de la santé qui traite la pandémie actuelle. C'est également le cas pour des termes très fréquents comme les articles, les pronoms ou les prépositions, qui en soi ne possèdent aucune signification pertinente dans le résultat. Ce type de mots fréquents reçoit donc un score TF-IDF très bas, ce qui rend ces mots négligeables dans le calcul de similarité entre les documents et donc dans la recherche.

En dernier point, supposons que $f_{m,d}$ est grand et que $f_{m,D}$ est petit. Alors $\log(|D| / f_{m,D})$ est relativement grand, et par conséquent m_d est également grand. En effet, les mots ayant un m_d élevé impliquent que m est un mot important dans d mais peu courant dans D. On dit que ce mot m a un fort coefficient de discrimination.

3.2.2 Approche "Bag Of Words"

Le modèle "Bag of Words" est une autre façon de vectoriser les données textuelles. On l'appelle un "sac" de mots, car toute information sur l'ordre ou la structure des mots dans le document est écartée. Le modèle se focalise uniquement sur la présence de mots connus dans le document, sans tenir compte de leur emplacement.

Cette approche est utilisée pour obtenir le nombre total de mots dans un texte. On se sert de cette méthode afin de créer une matrice de la composante de fréquence des termes, et le taux d'occurrence de chaque terme est obtenu en se servant de cette matrice. Ces fréquences de mots sont utilisées comme

caractéristiques pour former un classificateur *Silva and Ribeiro [2003]*. Cette méthode peut présenter certains inconvénients, tels que les 'Stop Words' (par exemple, "the") qui entraînent une augmentation du bruit dans l'analyse, ou l'absence de contexte. On utilise CountVectorizer de la bibliothèque 'scikit-learn'.³

3.2.3 Approche "Sentence-Bert : SBERT"

BERT *Devlin et al. [2018]* (Bidirectional Encoder Representations from Transformers) est une technique d'apprentissage automatique basée sur les Transformers pour le pré-entraînement du traitement du langage naturel (NLP) développée par Google.

BERT a été entraîné sur deux tâches : la modélisation du langage et la prédiction de la phrase suivante. Nous présentons le modèle qu'on a utiliser pour cette partie qui est Sentence-BERT⁴ (SBERT) (*Reimers and Gurevych [2019]*), une modification du modèle BERT utilisant des réseaux de "siamois" et de "triplets". L'architecture du réseau "siamois" permet entre autres d'obtenir des vecteurs de taille fixe pour les phrases d'entrée. En utilisant une mesure de similarité comme la similarité cosinus, les phrases sémantiquement similaires peuvent être trouvées.

Sentence-BERT, ou SBERT, utilise un encodeur qui peut convertir de longs extraits de texte en vecteurs. Dans un système de recherche sémantique de documents, SBERT génère un vecteur pour chaque document existant *Andrey [2021]*. Ces vecteurs sont ensuite indexés à côté du document original dans la base de données. Lors de la phase de déduction, SBERT convertit le nouveau document en un vecteur et le compare aux vecteurs de la base de données. La similarité entre deux vecteurs est proportionnelle à la similarité des documents qui les composent.

Pour cette partie, on a choisi la version "stsbert-large" du SBERT, il a un "STC benchmark" de 86.39%, qui est le référentiel de comparaison de similarité sémantique.

3.2.4 La similarité Cosinus : Discussion des résultats

Compter le nombre de mots communs ou encore calculer la distance euclidienne est l'approche générale utilisée pour faire correspondre deux documents similaires, qui se base sur le nombre de mots partagés entre les documents.

Cette approche ne fonctionne pas même si le nombre de mots identiques augmente alors que les documents traitent de sujets différents. Pour contourner ce problème, le cosinus est utilisé, en calculant l'angle entre les deux vecteurs.

Après avoir calculé la similarité en cosinus sur les données de référence (**Claim2**) en utilisant les différentes approches vues précédemment, nous allons essayer de comparer et d'évaluer les résultats en utilisant la précision, le rappel et la f-mesure. Nous écrivons un programme qui va compter les vrais positifs (VP), faux négatifs (VN), faux positifs (FP) et faux négatifs (FN) et les représenter dans des courbes qui montrent l'évolution et l'impact du seuil de classement sur nos scores, et ensuite trouver un seuil idéal pour le classement des textes suivant les différentes annotations.

On représente, dans ce qui suit, les courbes et résultats des métriques choisies. Ici, pour une meilleure visualisation, le seuil est multiplié par 100.

3. `sklearn.feature_extraction.text.CountVectorizer`. (s. d.). Scikit-Learn. Consulté le 21mai2022, à l'adresse https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

4. <https://github.com/UKPLab/sentence-transformers>

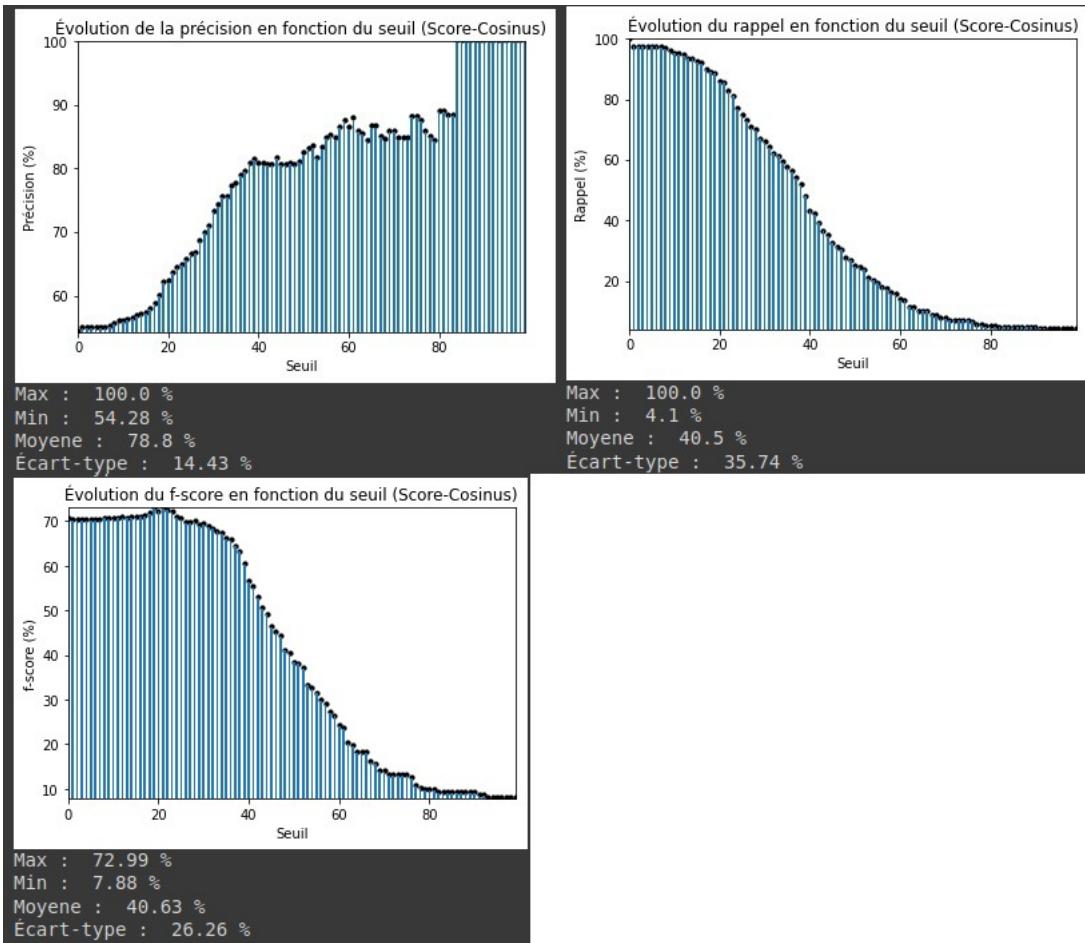


FIGURE 1 – Evolution de la *précision*, *rappel* et le *f-score* en fonction du Score Cosinus (seuil) pour CountVectorizer

Pour l'approche "Bag of Words" en utilisant CountVectorizer. Concernant la précision, on constate d'après les courbes, pour un seuil qui varie entre 0 et 40, qu'elle atteint les 80%. Cependant, pour l'intervalle de seuil qui varie de 40 à 80, on observe une croissance faible de la précision qui, atteint ensuite les 90% pour un seuil de 80.

D'après ces résultats, pour un seuil de 0 à 40, on peut dire qu'on a pu diminuer le nombre de faux positifs dans le classement. En revanche, on a le seuil = 80, qui est un seuil pour lequel on peut classer les textes qui sont similaires d'une manière efficace. Ensuite, pour le rappel, sur un seuil de 0 à 20, on constate une valeur élevée, ce qui signifie que les faux négatifs sont moins nombreux, on peut classer les textes similaires de façon efficace. Cependant, à partir d'un seuil = 20 et jusqu'à sa valeur maximale, on remarque que le rappel diminue de façon significative, donc on prédit des textes similaires alors qu'ils ne le sont pas.

Enfin, on ce qui concerne le f-score, à partir d'un seuil = 25, on constate qu'avec l'augmentation du seuil, le f-score diminue, et par conséquent, on n'arrive pas à distinguer entre les textes similaires et ceux qui ne le sont pas. Alors que, pour un seuil très restreint qui varie de 20 à 22, on voit qu'on peut classer les textes de manière efficace.

De la même manière, on analyse ces métriques pour la méthode TfIdfVectorizer et le modèle SBERT.

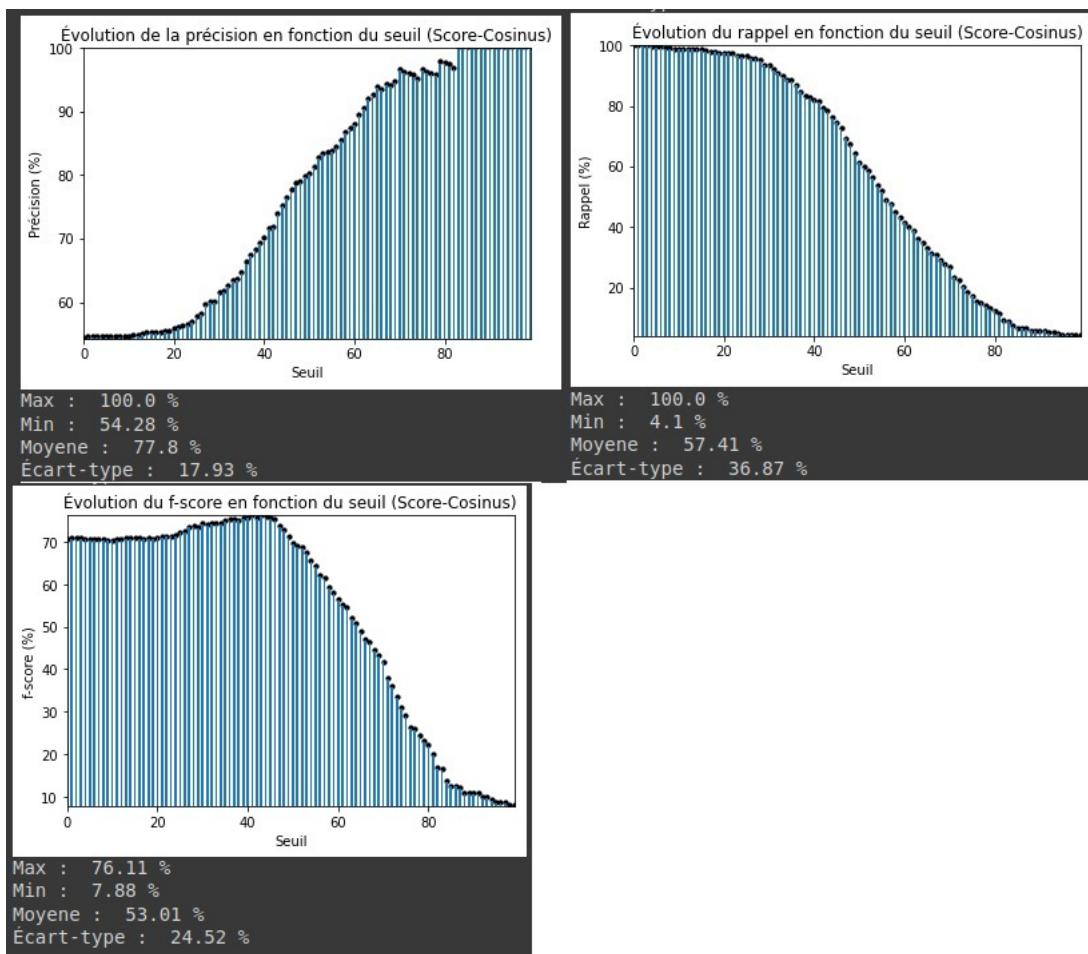


FIGURE 2 – Evolution de la *précision*, *rappel* et le *f-score* en fonction du Score Cosinus (seuil) pour SBERT

Concernant la précision, pour une variation du seuil de 0 à 20, on constate une légère variation de 54% à 56%. Ensuite, pour un seuil de 20 à 80, on remarque une croissance significative pour atteindre les 95% en précision. De ces résultats, et pour un seuil de 20 à 80, on a pu diminuer les faux positifs. Ensuite, à partir d'un seuil de 80, on peut classer les textes similaires d'une manière efficace.

Ensuite, pour le rappel, sur un seuil qui varie de 0 à 30, on a un bon résultat autour de 92%, ce qui signifie que les faux négatifs sont minimes. En revanche, à partir d'un seuil qui vaut 30, on remarque que la valeur de rappel diminue de manière significative, donc les textes similaires ne sont plus reçus comme tels.

Enfin, pour la courbe du f-score, on constate qu'à partir d'un seuil qui vaut 50, la valeur de f-score décroît. Cependant, en variant le seuil entre 20 et 50, on peut bien classer les textes.

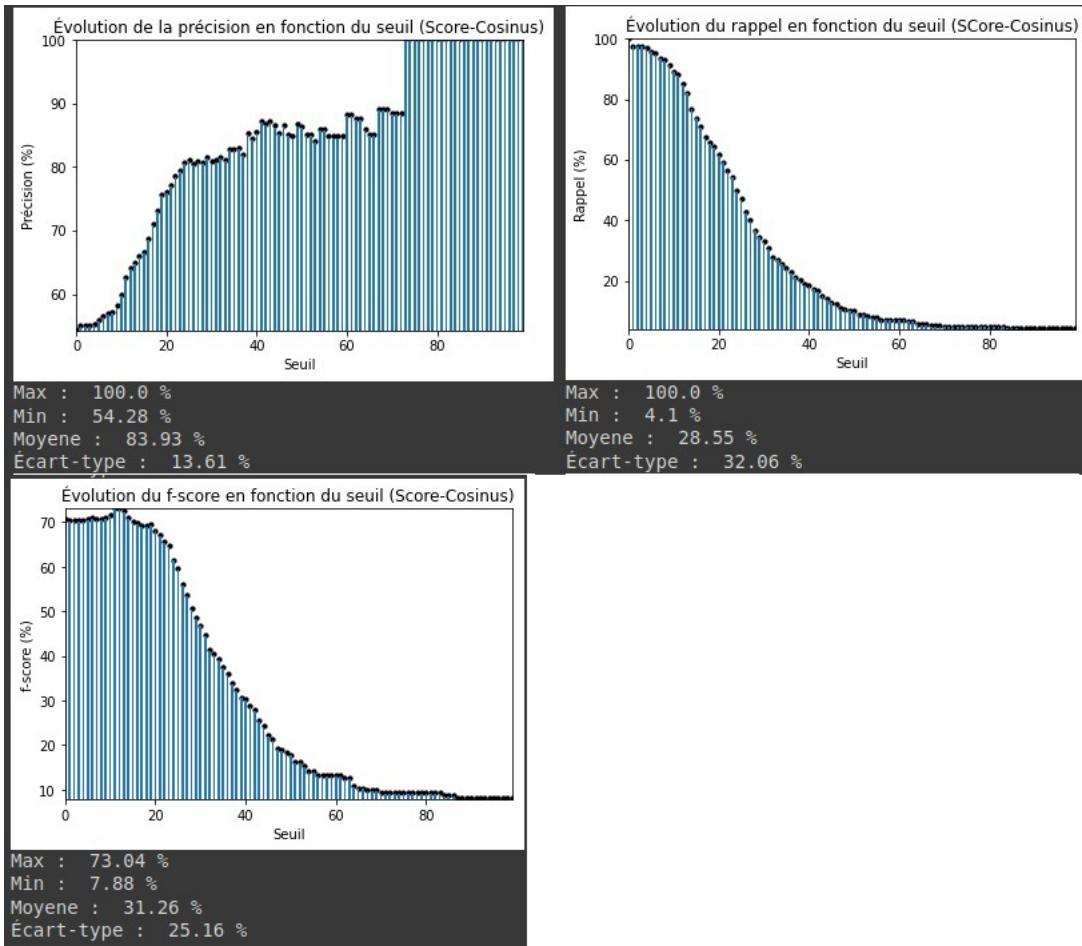


FIGURE 3 – Evolution de la *précision*, *rappel* et le *f-score* en fonction du Score Cosinus (seuil) pour TfIdfVectorizer

Suivant le même principe. Pour une variation du seuil entre 0 et 25, on remarque une croissance significative qui atteind les 80% en terme de précision. Ensuite, pour un seuil de 25 jusqu'à 70%, on a une légère fluctuation pour arriver à 88%.

D'après ces résultats, pour un seuil de 0 à 25, on peut dire qu'on a pu diminuer le nombre de faux positifs dans le classement. En revanche, on trouve que les meilleures résultats sont autour d'un seuil de 70. Pour la courbe du rappel, un maximum autour de 96% pour un seuil de 0 à 5. À partir d'un seuil de 5, on constate que la valeur de rappel diminue de manière significative. Pour finir, pour un seuil de 20, on remarque que le f-score est autour de 70% et commence à décroître significativement. On peut dire qu'autour d'un seuil de 12, on a les meilleurs résultats du f-score.

On regroupe ces résultats dans le tableau suivant :

TABLE 4 – Synthèse des résultats pour la précision, rappel et f-score

Approche	Précision		Rappel		f-score
	diminuer les FP	seuil (meilleur score)	diminuer les FN	seuil (meilleur score)	seuil (meilleur score)
Count	[0 - 40]	80	[0 - 20]	[0 - 20]	20
SBert	[20 - 80]	80	[0 - 30]	[0 - 30]	50
TfIdf	[0 - 25]	70	[0 - 5]	[0 - 5]	12

3.3 Comparaison des méthodes

Les deux approches utilisées ont donné des résultats très différents. Plus complexes dans leur implémentation, les méthodes syntaxiques se sont avérées globalement moins efficaces que celles basées sur la transformation en vecteurs.

Cependant, les méthodes syntaxiques utilisent de nombreux pré-traitement qui n'ont pas été appliqués lors des calculs des vecteurs. Il serait donc possible de combiner ces deux types de méthodes afin d'optimiser les résultats. De plus, l'analyse sémantique n'a pas pu être effectuée, mais les réseaux siamois de BERT ont permis de pouvoir prendre en compte cet aspect.

Si ces combinaisons n'ont pas pu être effectuées, les méthodes ici présentes ont cependant servies à mettre en œuvre d'autres besoins du projet. La vectorisation permet de faire de l'analyse par sujets, partie 4, et le calcul des cosinus sera utilisé en partie 5 afin de faire une base de données. Les méthodes de traitement vues pour l'analyse syntaxique sont aussi étudiées pour le filtrage en partie 5, où elles se montreront plus adaptées pour créer un moteur de recherche que pour directement annoter des couples de textes.

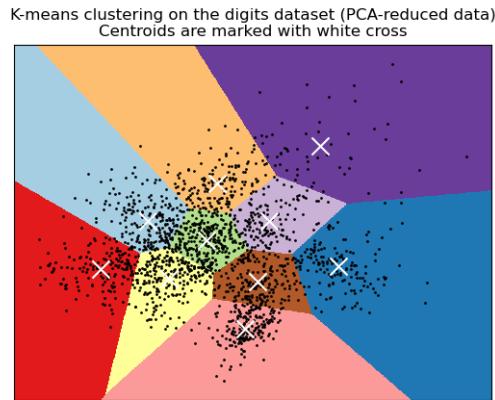


FIGURE 4 – Visualisation du fonctionnement de KMeans, *scikit-learn*

4 Regroupement par sujets

Les approches précédentes visent généralement à trouver l'association la plus proche entre deux textes. Dans ce projet, le but est de l'appliquer aux bases de "Claims", d'affirmations, et de commentaires postés sur Twitter. La complexité de cette tâche se retrouve dans le nombre important de combinaisons possibles. Avec nos échantillons de 10^3 claims et environ 8×10^6 Tweets, le nombre total de combinaisons possibles est de l'ordre de 10^9 . Même en supposant une puissance de calcul permettant l'obtention d'un résultat en quelques secondes, et une précision dépassant les 90%, il reste difficile de se satisfaire d'une seule et unique réponse, car il est possible qu'un même claim corresponde à plusieurs tweets.

En considérant l'ensemble des bases de données comme un corpus de texte, composés de plusieurs entités uniques qui sont les claims d'un côté et les tweets de l'autre, il est possible de l'assimiler à un document complexe. Cette approche a notamment été utilisée sur des commentaires postés sur twitter, dans le but de discerner les différents sujets de discussion autour de la vente de logiciel ([Lu et al. \[2011\]](#)). La recherche de sujet dans les textes constitue un défi du fait de la complexité des documents, mais aussi à cause de la notion de sujet elle-même.

Il est possible de regrouper les phrases de plusieurs manières, car un segment de texte ne s'associe pas forcément avec un seul et unique sujet ([Stanik et al. \[2021\]](#)). S'il est possible de dire, par exemple, qu'un tweet est relié au sport, ce même tweet peut aussi faire référence au monde politique. De plus, les sujets peuvent eux-mêmes regrouper des "sous-sujets", en allant à un niveau de détail plus important. Pour reprendre l'exemple du sport, il peut s'agir spécifiquement de football, voir même de football européen, ou même très exactement de la seconde division de la ligue espagnole.

Regrouper les phrases similaires entre elles offre donc l'avantage de proposer une vision d'ensemble. La capacité à faire un ensemble cohérent permet à l'utilisateur finale d'avoir une compréhension plus profonde des données, qui pourrait d'ailleurs se faire sur plusieurs niveaux, en répétant l'opération afin d'avoir des groupes de taille de plus en plus réduite.

Deux techniques seront ici abordées afin de réaliser cette tâche. Il s'agit ici de faire de la classification non supervisée, en utilisant des méthodes de *clustering* pour l'analyse de sujets. Ces algorithmes seront capables, sans information à priori, de séparer les différentes valeurs en groupes. Le "*K-Means*" , ou K-Moyennes, utilise des valeurs numériques ([Grira et al. \[2004\]](#)). L'allocation latente de dirichlet, ou *LDA*, est particulièrement utilisée pour les corpus de texte.

4.1 Regroupement par K-Moyennes

Les algorithmes de type *K-Means* séparent les données numériques passées en entrée de l'algorithme en K groupes. Après avoir fait une vectorisation, l'algorithme va déterminer quels sont les centres des groupes que l'on cherche à déterminer (*sujets ici*), qu'il est aussi possible de passer en paramètre [Wang and Su \[2011\]](#). Il s'agit donc à partir de là d'assigner un centre, ou centroïde. La démarcation entre les différents groupes va à la fois dépendre de la distance aux centroïde et des centroïdes voisins, comme il est possible de le voir sur la *Figure 4*, fournie directement par la bibliothèque Scikit-Learn et montrant le fonctionnement dans un espace à deux dimensions. L'algorithme va chercher à déterminer un point central optimal μ tel qu'il va satisfaire l'équation :

$$\sum_{i=0}^n \min_{\mu_j \in C} (||x_i - \mu_j||^2) \quad (2)$$

pour n points dans un groupe, avec un nombre K de groupes choisis. Ces groupes ne sont pas forcemment de tailles homogènes, comme il a été observé dans le projet. Il est important de noter que le point qui va servir de "centre" au groupe ne correspond pas forcemment avec les coordonnées d'un point fournit en entrée à l'algorithme.

Dans le cas d'utilisation traité ici, chaque groupe associé à un centroïde sera interprété comme étant un sujet commun aux textes, claims ou tweets, qui lui sont associés. Ici, les coordonnées des vecteurs correspondent aux fréquences d'apparition de chaque mot, pondérées par l'inverse de la fréquence d'apparition du mot dans l'ensemble du texte dans le cas de la *TF-IDF*.

Les différents sujets sont donc associés à des combinaisons de mots retrouvées dans plusieurs phrases, et contrairement aux méthodes précédentes, les phrases ne seront pas mises en opposition deux-à-deux car le résultat sera obtenu en prenant en compte l'ensemble du groupe de mots sélectionné. Cet ensemble peut être l'intégralité des données sélectionnées, mais les phrases utilisées peuvent aussi être pré-sélectionnées en fonction d'autre critère. Par exemple, il est possible de définir à priori un certain seuil de similarité par rapport à une phrase, ou encore certains mots qui doivent apparaître dans les textes sélectionnés. Ces possibilités seront discutées dans l'application des algorithmes, vue en *partie 5*.

Pour une première implémentation, et afin d'avoir une référence pour évaluer la pertinence des sujets ainsi fondés, les données *Claims2*, avec des associations de deux claims A et B, seront utilisés. L'ensemble de phrase est la concaténation des claims A et B. Après vectorisation de l'intégralité du datafram, il est possible de passer la matrice en argument afind d'obtenir les clusters. La variable ici sera le nombre de clusters. Il s'agit d'évaluer, pour cet exemple, comment va évoluer la précision, le rappel, et le f-score en fonction du nombre de sujets sortis par le *K-Means*. Pour deux Claims A et B, la correspondance entre leurs allocations respectives et les annotations faites dans la base de donnée servira de vérité de terrain.

L'algorithme utilisé est le K-Means de *Scikit-Learn*. Bibliothèque de référence en machine learning, elle permet en plus de divers outils d'analyse et traitement de données déjà utilisé comme la vectorisation par *TF-IDF* et le traitement de texte l'implémentation de multiples algorithmes de classification, supervisée ou non. À chaque phrase sera associé un vecteur, et il sera assigné à un centroïde. Sont considérés comme des résultats positifs des claims A et B avec le même centroïde. Si l'annotation était E, E* ou ST, il s'agit d'un vrai positif.

En suivant ce principe, il est possible de calculer les vrais positifs, faux positifs etc... et d'obtenir un score de précision et de rappel, puis les combiner pour avoir le f-score. Afin de pouvoir juger de la composition des différents sujets, pour chaque label est affiché à la fois le nombre total de phrases, le nombre de phrase par partie de document, *A ou B*, ainsi qu'un échantillon aléatoire de n phrases visible sur la figure 15 en Annexe. Dans cet exemple précis, ces sujets sont centrés autour des personnalités nommées dans les phrases.

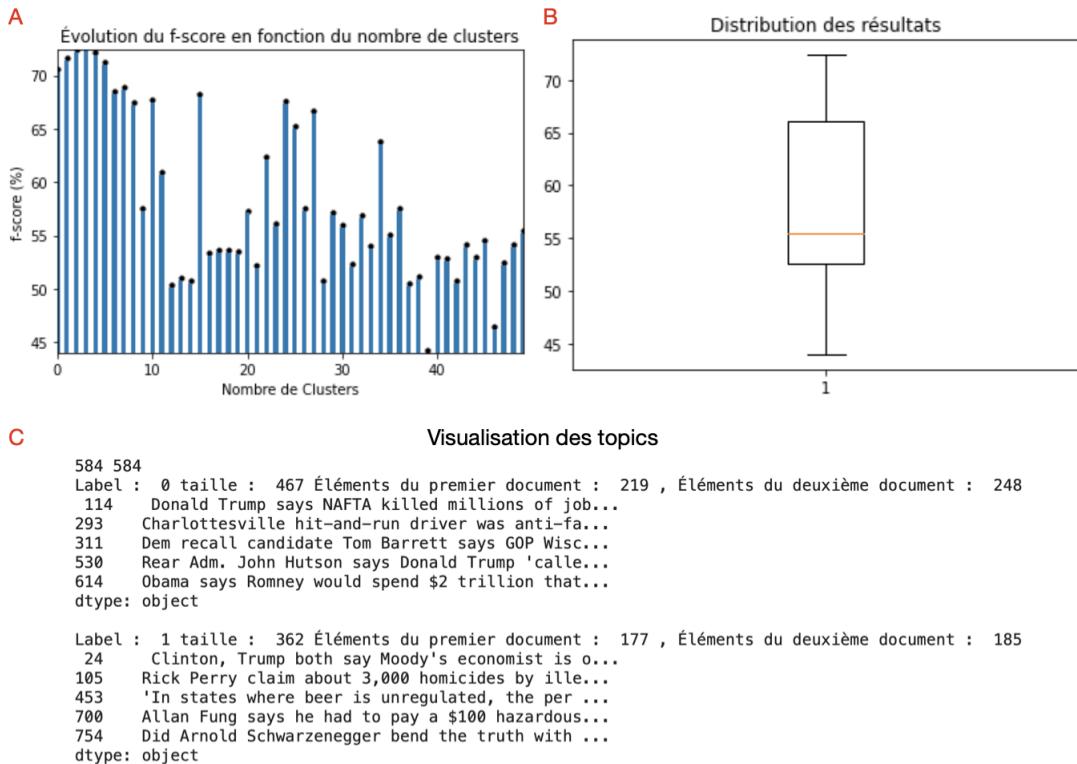


FIGURE 5 – Résultats obtenus avec **K-Means** - **A.** Graphique représentant le f-score $\times 100$ en fonction du nombre de sujets - **B.** Distribution des résultats pour l'ensemble des 50 tests - **C.** Visualisation des deux premiers topics pour n=10

Les résultats de la figure 5 montrent l'évolution du f-score, ici en pourcentages, quand le nombre de centroïdes varie entre 1 et 50 inclus, ainsi que la distribution des résultats. Dans cet exemple, et cette tendance se confirme sur plusieurs essais, le f-score est maximisé pour un nombre de sujets compris entre 0 et 10, mais aussi entre 20 et 30, où on retrouve des résultats supérieurs à 65% de correspondance avec les annotations, pour les 584×2 claims disponibles.

Il faut cependant considérer que la notion d'appartenance à un sujet reste non seulement relativement subjective, mais ne donne pas l'information sur le sujet qui lie les deux phrases. Ainsi, deux phrases peuvent chacune être correctement placée dans un sujet qui les correspond, mais donner un résultat négatif dans la mesure. Il est aussi préférable d'avoir un plus grand nombre de sujets, à précision égale. Celà signifie que la clusterisation est d'autant plus performante, permettant à la fois de garder les associations entre phrases, ici les claims, et une vision détaillée de la distribution du document.

En adaptant la méthode de K-Means à un ensemble de textes il est donc possible d'extraire un nombre K de sujets dans lesquels seront placées les phrases. Si la valeur de K semble dépendre de la composition du texte, il est intéressant de voir que même avec un nombre élevé il est possible de garder une cohérence entre le résultat sorti par l'algorithme et ce qu'un observateur peut obtenir en classant les phrases comme c'est le cas sur la base Claims2. Si la correspondance n'est pas parfaite, les résultats démontrent bien que la classification permet de donner du sens à un ensemble de textes. Il existe cependant une autre méthode plus couramment utilisée pour l'attribution de sujets, l'allocation latente de Dirichlet.

4.2 Allocation Latente de Dirichlet

L'allocation Latente de Dirichlet, ou *LDA* en anglais, est une méthode de référence pour l'analyse de textes. Les textes sont des données volumineuses et complexes, qu'il peut être difficile de visualiser dans leur ensemble. Les méthodes utilisant le principe de la LDA ont l'avantage de pouvoir être utilisées pour de

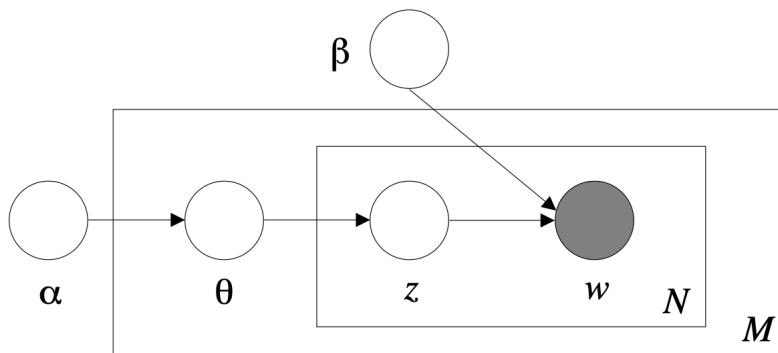


FIGURE 6 – Représentation graphique de la LDA. La boîte M représente les M documents, la boîte N représente les N topics. θ , tiré de α donne une distribution z de topics, et β de mots, permettant de choisir un mot w [Blei et al. \[2003\]](#)

nombreuses applications dans l’analyse de texte, que ce soit pour l’attribution de d’auteurs, ou comme ici pour générer des modèles de sujets. Avec l’accroissement de la quantité de données textuelles disponible, comme vu ici avec Twitter, il est nécessaire de développer des outils capables de faire apparaître une organisation dans un ensemble autrement non structuré, comme c’est le cas pour toutes les bases de données utilisées dans le cadre de ce projet [Rieger et al. \[2020\]](#).

La LDA est un modèle probabiliste de génération de corpus. Il s’agit de partir du principe que les documents sont un ensemble de sujets latents, c’est-à-dire dont l’existence même est diffuse au sein de ces documents. Ces sujets sont eux-mêmes constitués d’une association de mots, caractérisés par leur distribution au sein de ces sujets. L’idée est que, en fonction de la présence d’un mot dans le document, et ce associée à celle de d’autres mots, il est possible de pouvoir en déduire son appartenance à un sujet.

Dans l’exemple pris pour ce projet, les différents claims et tweet font donc office de documents, comme cela a déjà été fait sur des commentaires par [Albalawi et al. \[2020\]](#). Le mot est l’unité de base, représentée pour l’algorithme par un vecteur dont la seule composante non nulle est à l’indice i , du i -ème mot dans l’ensemble de taille N qu’est le document. Le corpus, lui, est une collection de D documents. Le corpus est généré selon une probabilité :

$$p(\theta, z, w | \alpha, \beta) = p(\theta, \alpha) \prod_{n=1}^N p(z_n | \theta) p(w_n | z_n, \beta) \quad (3)$$

où les variables $z_{d,n}$ représentent le topic choisi pour le mot $w_{d,n}$, θ_d représente la distribution de topics dans le document d’indice d , α et β définissent les distributions à priori sur θ et β respectivement, et où β_k décrit la distribution du topic k .

Pour générer le corpus, l’algorithme choisit θ tel que θ suive une distribution de Dirichlet paramétrée par α . α est généralement un vecteur de dimension K , K le nombre total de sujets. Classiquement, dans une distribution de Dirichlet, le paramètre α va indiquer si les éléments sont distribués équitablement si $\alpha = 1$, s’ils sont éloignés dans l’espace pour $\alpha > 1$, ou si au contraire ils sont regroupés avec $\alpha < 1$.

Une fois θ obtenu afin de caractériser la distribution l’ensemble des sujet dans le corpus documents, pour chaque mot w_n est choisi un sujet z_n suivant une distribution Multinomiale qui est elle paramétrée par θ . Le mot du sujet sera choisi en suivant une distribution multinomiale paramétrée par β_k , avec $k = z_n$ le sujet qui lui correspond [Bietti \[2012\]](#). Le but final est ici de générer un corpus ressemblant le plus possible à l’ensemble de texte original, ce qui sera associé à une attribution de sujets cohérente avec le "sens" du texte.

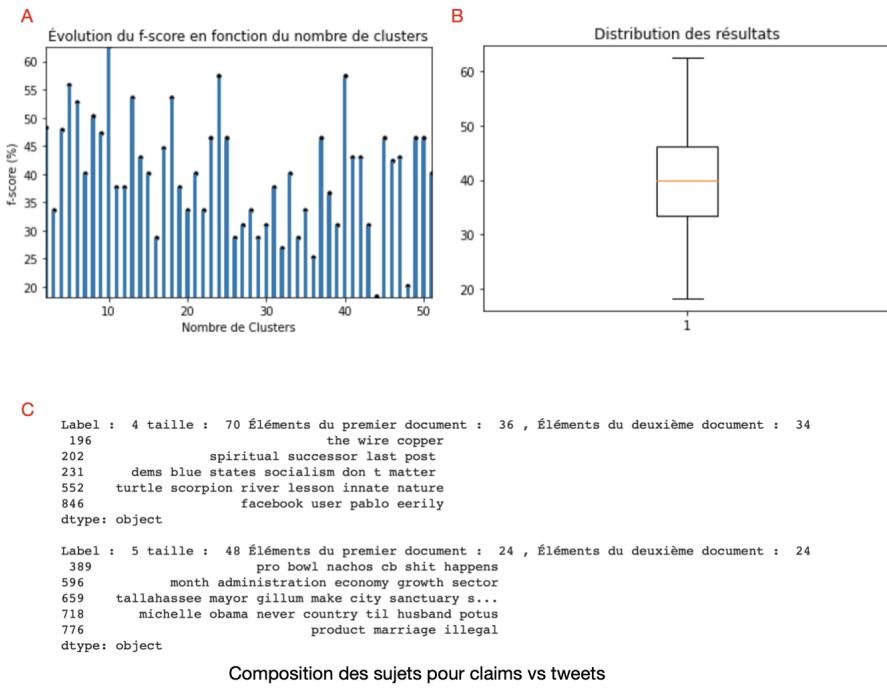


FIGURE 7 – Résultats obtenus avec la **LDA** - **A.** Graphique représentant le f-score $\times 100$ en fonction du nombre de sujets - **B.** Distribution des résultats pour l'ensemble des 50 tests - **C.** Visualisation de deux topics sur Claims et Tweets

La bibliothèque Gensim propose un ensemble de méthodes pour implémenter la LDA sous Python. Afin d'obtenir un modèle, l'algorithme prendra en argument le corpus de texte, qui sera ici une partie ou l'ensemble de la base de données considérée, ainsi que le dictionnaire de mots, et enfin le nombre attendu de sujets (à l'image de K Means). Le corpus et le dictionnaire sont tout deux créés avec des fonction de Gensim. [Gupta et al. \[2022\]](#) Une attention particulière est donnée ici au traitement de texte, afin de maximiser les chances d'obtenir des sujets cohérents.

Un des avantage de la LDA est qu'elle permet une visualisation des mots qui composent un sujet, et de leur importance dans la construction du sujet, comme vu en annexe sur la figure 16. Celà permet notamment de visualiser les mots qui n'ont pas une importance significative dans la compréhension et la segmentation des textes, et pouvant être supprimés lors du traitement de données.

De manière analogue au K Means, il est possible de créer un modèle basé sur des données de référence. Celui-ci sera doté d'un ensemble de K sujets, eux-mêmes composés de mots. La principale différence avec le K Means est qu'au lieu d'affecter à chaque document un sujet, le modèle de LDA créé va affecter une probabilité d'appartenance à chaque sujet. Pour sortir un résultat similaire à celui proposé par K Means, il suffit cependant de prendre, pour chaque document, le sujet qui coincide avec la plus forte probabilité au texte. Ainsi, il est possible encore une fois de vérifier quantitativement et qualitativement la correspondance entre le modèle, et les annotations d'une base de données.

Sur la figure 7, les résultats sont globalement moins bons que pour le K Means, avec un score inférieur d'environ 10% sur l'ensemble des résultats. Cependant, il est intéressant de noter que les tendances sont similaires. Encore une fois, les résultats sont les meilleurs quand sont choisis entre 20 et 30 sujets, ou s'il y en a moins de 10. La LDA permet donc de détecter les éléments pourtant être décrits comme appartenant au même sujet avec précision plus faible que le K Means.

Si la LDA offre, dans la tâche considérée, des résultats moins bons que le K Means pour une implémentation et un fonctionnement plus difficile à appréhender, elle n'en reste pas moins une méthode plus adaptée aux textes, et permettant une visualisation plus complète de la distribution des mots dans les sujets, et

des sujets dans le corpus. Ainsi, il serait possible dans la modélisation de sujet pour les claims et tweets de faire usage à la fois du K Means et de la LDA, mais pour réaliser différentes tâches.

4.3 Vers l'application

Le projet vise à terme à proposer plusieurs solutions afin de parcourir simultanément les bases ClaimsKG et TweetCov19, dans le but de lier par exemple un ou une série de tweets avec un ou un ensemble de claims. Dans cette optique, les méthodes de regroupement par sujet auront chacune leur utilité et leur application.

En faisant des liens entre les bases, il est possible d'aller dans une direction ou l'autre, c'est-à-dire de partir d'un tweet, pour trouver un claim et vis-versa. Quand une recherche est faite, l'utilisateur peut se voir proposer des éléments similaires. Dans ce cas, il est possible d'utiliser le KMeans, par exemple, et de trouver un groupe de tweets qui serait labellisé comme celui recherché par l'utilisateur. Celà permettrait, en partant d'une recherche d'aller trouver des éléments similaires, et potentiellement étoffer le nombre de résultats obtenus par la suite.

Lorsqu'une recherche est effectuée, si le désir de l'utilisateur est de trouver plusieurs éléments similaires, la LDA en particulier offre plusieurs possibilités. Si le KMeans et la LDA permettent tout deux de sélectionner le nombre de sujets créés, et donc quelque part de pouvoir sélectionner à quel point la clusterisation des données sera *fine*, donc à quel point les groupes peuvent être petits, elle va aussi renvoyer la probabilité d'appartenance à un sujet pour tous les N sujets. Il est donc possible de traiter la problématique du début, qui était le fait qu'un même élément de texte puisse correspondre potentiellement à plusieurs sujets.

En combinant le KMeans qui s'est prouvé plus fidèle que la LDA, il est possible de regrouper le texte recherché avec des textes similaires, et ensuite d'appliquer la LDA afin de trouver, dans ce groupe, tous les sous-groupes de textes auxquels le texte de départ peut potentiellement appartenir. Ces méthodes offrent donc des combinaisons d'utilisation, et se retrouvent très utile dans le cadre du projet. S'il n'a pas été possible d'implémenter une version "finale", l'analyse de ces deux méthodes permet de mesurer à quel point l'apprentissage supervisé peut donner des résultats fidèles aux observations sur les données, mais aussi de pouvoir entrevoir plusieurs scénarios d'utilisation et des réponses adéquates basées sur ces outils.

A

Username	Timestamp	Followers	Friends	Retweets	Favorites	Entities
234fe4a19cc1a3336095fb3780bcc1	Mon Sep 30 22:00:37 +0000 2019	619	770	0	0	null;
4592f39636d87af8fb4b17b7e2e4c0	Mon Sep 30 22:01:24 +0000 2019	36365	19344	13	17	nazi:Nazism:-2.742538749414189;blood money:Blo...
5d1888dd974fa4a8679c25e2ead03	Mon Sep 30 22:06:21 +0000 2019	5018	1933	0	0	vaccine:Vaccine:-2.6651530673745762;anti vaxxe...

B

text	date	truthRating	ratingName	author	headline	named_entities_claim	named_entities_article	keyword
Malia Obama cashed a \$1.2 million tax refund c ...	Unknown	-1	OTHER	Unknown	Did Malia Obama Cash a \$1.2 Million Check?	Facebook,Fan Fiction,Junk News,Malia Obama,Sno...	Malia Obama	Na
High diver is saved from jumping into a drain...	Unknown	-1	OTHER	Unknown	High Diver Saved By Cross	Cincinnati Post,Islam,Scripture lesson,Univers...	shadow on the wall	ASP Artic

FIGURE 8 – A. Dataframe de tweets B. Dataframe de claims, comportant des formattages différents

5 Modules complémentaires

L'étude de l'association de texte, ensuite du regroupement par sujet, permet d'obtenir plusieurs outils de base afin de réaliser le traitement de texte. Ces éléments peuvent notamment à la fois servir dans le cadre d'une recherche par un utilisateur, mais aussi pour réaliser les tâches nécessaires à la production d'un outil de recherche. Cette partie propose de répondre à deux problématiques de mise en place de l'outil final, qui sont la création d'une base de données de référence basée sur ClaimsKG et TweetsCov19, et le filtrage d'éléments.

5.1 Crédation d'une base Claims-TWeets

Les tests effectués jusque là, pour fournir un indice de performance sur les méthodes utilisés, se sont basés principalement sur la base Claims2. Elle offre la possibilité de faire de la comparaison de textes à plusieurs niveau, grâce aux multiples annotations, et se prête donc parfaitement au cadre du projet. Cependant, l'objectif reste de lier les bases ClaimsKG et TweetCov19. Il est donc important d'arriver à obtenir une réalité de terrain sur ces bases, non seulement pour essayer les modèles mais les paramétriser. Différents jeux de données peuvent avoir différentes distribution, donc il est nécessaire d'utiliser des données ressemblant le plus aux cas "réels" du projet, tweets et claims.

La principale différence avec les textes précédemment utilisés se retrouve dans le format. Si Claims2 est construite avec des données de ClaimsKG, formatées de la même manière, TweetCov19 est une base de donnée totalement différentes. En chargeant les dataframes, la différence la plus marquante réside dans le fait que la base de tweets ne contient pas les textes intégraux. Les données sont formatées et ne renvoient que des entités, associées au score de la métrique utilisée, comme visible sur la figure 8.

Pour que les données soient comparable, il faut donc les traiter de manière à ce qu'elles se trouvent sous une forme similaire. Ici, le K Means en particulier permet de vérifier si les éléments des deux bases se retrouvent dans des clusters similaires, ce qui veut dire qu'il est possible de faire un lien. Si au contraire les données se retrouvent systématiquement dans des labels dépendant des bases de données d'origine, il ne serait pas possible de les traiter comme des paires, ou des documents du même corpus.

Afin d'enregistrer les données dans un fichier CSV, il est important de prendre en compte le fait qu'un

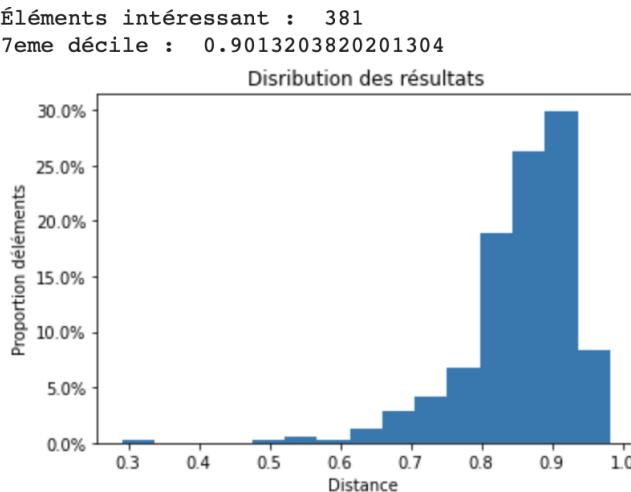


FIGURE 9 – Distribution des claims-tweets testés en fonction de leur distance

tweet et un claim tiré au hasard n'aient que très peu de chances de correspondre à un sujet similaire. Après avoir enlevé les entités nulles, il ya 5×10^6 tweets et 10^3 claims, donc 10^9 combinaisons. En annotant chaque couple, le jeu de données serait très déséquilibré, et beaucoup de temps serait perdu sur des annotations redondantes. Il est donc nécessaire de pouvoir filtrer en amont les éléments qui ont peu de chances de se correspondre.

L'approche a été de se référer aux méthodes de calcul de distance vues précédemment. Elles ont l'avantage de permettre un calcul rapide, et elles sont plus fiables que les méthodes basées sur le clustering. Toutefois, il est important de choisir un seuil pertinent, qui soit assez haut pour que les éléments soient "proches", mais assez bas pour qu'il y ai une représentation de l'ensemble des éléments. Idéalement, ce seuil devrait être similaire à celui qui sera utilisé par l'utilisateur lorsqu'il fera une recherche.

Les distances ont été calculées sur un échantillon aléatoire de claims et tweets traités afin que la LDA les place dans des groupes similaires. Le calcul de la distance se fait ici avec la *TFIDF*, en utilisant la mesure du *cosinus*. Dans cet échantillon, plusieurs éléments renvoient une erreur de calcul, et ce même après avoir supprimé les claims "vides", après traitement. En prenant 100000 couples aléatoire, en moyenne seuls 0.5 % donnent un résultat qu'il est possible d'interpréter. Toutefois, cela suffit à dégager une tendance, car la base Claims2 par exemple est constituée de 500 valeurs.

En figure 9, l'algorithme renvoie que 30 % de ces valeurs ont un cosinus supérieur à 0.9. En prenant automatiquement le 8ème décile du test, il est possible de filtrer 80 % des couples claims-tweets, en partant du principe qu'ils ne sont de toutes façons pas assez similaires. Cette valeur a été choisie pour "tester" l'algorithme permettant de créer la base.

Une fois le seuil choisi, l'algorithme construit pour constituer la base va donc itérer des comparaisons sur des couples claims-tweets, de manière à vérifier s'il n'existe pas déjà dans la base de données le même couple, et ensuite si le cosinus est bien supérieur à la valeur de référence l'élément sera ajouté à la base de données, suivant la structure ci-dessous :

Il a été ainsi possible de produire un premier exemple de base de données. Cependant, les bases sont souvent revues plusieurs fois afin que les annotations soient validées, et que la base contienne le moins de bruit possible. Au cours des différents essais, la proportion de textes faisant partie du même sujet était entre 20 et 30 % après filtrage. Cependant, ce nombre peut varier en fonction de la définition d'un sujet.

Les annotations ressemblant à celles de Claims2 ne permettent pas non plus de dire si les claims appartiennent tous deux à un topic très général, par exemple la politique, ou si au contraire les textes sont très proches avec dans cet exemple, les élections pour la mairie du 8e arrondissement de Paris. Il serait intéressant de faire une étude plus approfondie sur ce sujet en particulier, cependant nous n'avons pas pu aller plus en

Data: Base de données,Tweets,Claims,seuil
Result: Base de données

```

int count ← 100000;
String Annotation;
for (i=0,i<count,i++) do
    Tweet=ChoixAléatoire(Tweets);
    Claim=ChoixAléatoire(Claims);
    if (( CouplePasDansLaBase(Tweet,Claim)) and (Cos(Vector(Tweet),Vector(Claim))>seuil ))
        then
            Annotation ← Annoter(Tweet,Claims);
            Base.Ajouter(Tweet,Claim,Annotation);
        end
    end
end
```

Algorithm 1: Algorithme de construction de la base

détail sur cet aspect.

5.2 Filtrage des textes

« Le passage du filtrage de corpus textuels volumineux par mots-clés (word spotting) au filtrage par thèmes (topic spotting), améliore le recueil d'informations bien ciblées. » [Lelu \[2002\]](#).

Dans un premier temps, nous avons extraits les indicateurs temporels des tweets et des claims qui exerce une grande influence sur la pertinence des résultats puisque celle-ci permet de les contextualiser. Ainsi, on peut déterminer à partir du mois et de l'année (le jour étant beaucoup trop précis) les différents claims correspondants à un tweet. Par exemple pour le tweet 1, le filtrage énonce 15 claims datant de la même année et du même mois.

Dans un second temps, nous avons filtré les résultats par Noms Propres permettant de définir si les phrases parlent des mêmes personnes. En effet, le filtre détermine pour un seul tweet tous les claims qui ont plus de la moitié des Noms Propres en communs. Par exemple, pour le tweet 1 on retrouve un seul et unique résultat (claim 215) qui correspond parfaitement. Le tweet 1 que l'on rappelle étant « «Trump needs to immediately divest from his businesses and comply with the emoluments clause. Iran could threaten Trump hotels *worldwide* and he could provoke war over the loss of revenue from skittish guests. His business interests should not be driving military decisions. — Ilhan Omar (@IlhanMN) January 6, 2020» et le claim 215 étant « In January 2020, U.S. Rep. Ilhan Omar advised Iran to attack Trump-branded hotels in the world, thus committing treason. ». Le filtrage pour ce premier tweet a donc été très précis.

Pour finir, nous avons écrit un programme pour pouvoir tester les différents filtres avec les différentes approches textuels que l'on a appliquée au premier tweet. Ce premier programme calcul par système de point la similarité. Il consiste à :

- Supprimer les mots inutiles des claims et du tweets que l'on veut faire correspondre.
- Ajouter 5 points si l'année correspond, 10 points si le mois correspond et 15 points si le jour correspond.
- Ajouter 20 points si une correspondance de plus de 50% des Noms propres
- Ajouter 50 points si la distance de Levenshtein est plus grande que 30 et que le ratio est plus grand que 0.6. Le programme ne donne comme résultat seulement les résultats ayant plus de 40 points.

Ainsi pour le premier tweet, les claims correspondants sont : 215,690. On retrouve le tweet 215 qui correspond parfaitement avec le tweet 1.

Cette méthode peut donc être développée d'avantage et proposée afin de créer des liens entre des tweets et des affirmations, car elle offre déjà une perspective de résultats satisfaisants.

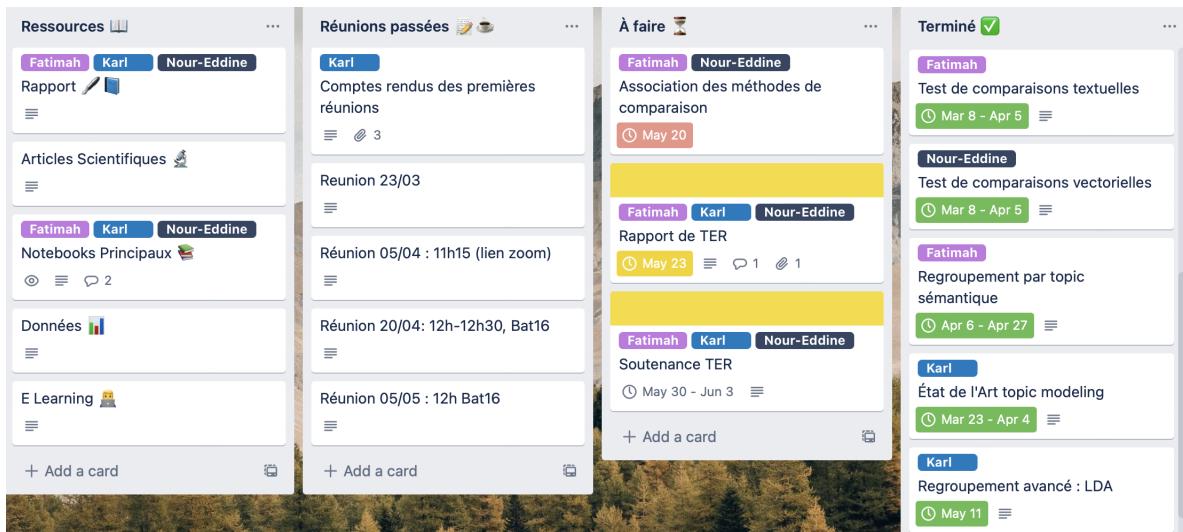


FIGURE 10 – Plateforme de gestion du projet, avec backlog et ressources (*Trello*)

6 Gestion de Projet

Nous proposons dans cette partie de faire un retour, après la présentation des résultats, sur l’organisation ayant permis de maximiser la performance de l’équipe en vue de la soumission de ce projet, qui ne s’est pas avéré sans défis.

Le T.E.R a été organisé autour d’un groupe de trois, Fatimah, Karl et Nour-Eddine, encadré par Mr Todorov. Le suivi s’est fait sur la base de réunion bi mensuelles. Aux vues de l’effectif du groupe et de la nature de la tâche, une organisation agile a été préférée, avec Karl dans le rôle de Scrum Master et Product Owner. Dans ce sens, les réunions avec l’encadrant servaient à juger de la correspondance entre résultats et attentes, et de fixer des objectifs futurs au travers des échanges avec le product owner.

Selon le framework scrum et suivant la fréquence des réunions, le projet s’est organisé sous forme de sprints de deux semaines. Ceci permet de viser une évolution constante des résultats, tout en donnant la possibilité d’ajuster le travail au fur et à mesure, sans avoir une ligne directrice rigide. Celà s’est retrouvé utile notamment car si la base Claims2 a été choisie comme référence, ça n’a été qu’après des tests de plusieurs bases de données, dont le format a posé des problèmes et ensuite entraîné des retards. En ajustant les objectifs et en redistribuant le travail, il a été possible de réaliser un grand nombre de tâches malgré celà. Un point d’honneur a été mis sur la communication entre les membres du groupes, avec des retours pour prendre mesure de l’avancée mais aussi être capable d’harmoniser les résultats.

Nous avons choisis de faire les tâches en parallèle, et non concentrer toutes les "forces" de l’équipe sur une seule. Celà permet d’éviter de perdre trop de temps sur une tâche si elle s’avère plus compliquée que prévue, quitte à l’abandonner. C’est ce qu’il s’est passé pour l’exploitation des bases de données au début, qui s’est avéré problématique. Les deux autres membres du groupe avançaient, donc il a été possible de minimiser le temps perdu. Au final, Fatimah s’est occupée des tâches de traitement de texte, Nour-Eddine de celles en rapport avec la vectorisation, et Karl du regroupement par sujet et de la création de la base.

Afin de garder un suivi du projet, et permettre un accès rapide et facile à l’ensemble des ressources du projet, la plateforme Trello (figure 10) a été utilisée car elle permet à la fois d’afficher clairement le backlog du sprint en cours, l’incrément produit avec les tâches effectuées, leurs dates et leur affectation, et elle propose aussi une vue plus générale avec une feuille de route (figure 18). Il est aussi possible d’y centraliser toutes les ressources.

Cette organisation a permis des échanges fluides, une adaptation constante, et de se concentrer sur les tâches à réaliser tout en gardant une vue sur les objectifs.

7 Conclusion

Le projet traité dans ce rapport visait à créer des liens entre une base de données d'affirmations vérifiées par des organismes experts, ClaimsKG et des tweets d'utilisateurs quelconques, dans TweetsCov19. Pour cela, il y avait notamment deux défis principaux qui se sont offerts à nous, à savoir la correspondance d'un couple de texte, et l'analyse de sujets.

Pour la correspondance de textes, des méthodes basées sur la syntaxe des textes ont été opposées à d'autres méthodes basées sur leur fréquence d'apparition. Elles ont toutes deux utilisées diverses métriques servant à mesurer la "distance" entre les textes. Cependant, si les méthodes numériques ont bien permis d'observer une tendance dans le choix des seuils, les résultats pour les méthodes syntaxiques n'ont pas été concluants. Ces méthodes ont cependant pu être réutilisées.

Concernant l'analyse de sujets, qui consiste à créer des groupes de claims et de tweets similaires, la méthode de classification non supervisée KMeans s'est opposée à l'allocation latente de dirichlet. Si la première méthode s'est avérée plus fidèle que la première aux annotations, elles ont toutes deux montré la même tendance en termes de nombre optimum de topics pour le corpus de la base de référence, claims2. La LDA propose cependant une analyse plus fine, et multithread, même si elle n'a pas été explorée spécifiquement ici.

Ces méthodes ont cependant permis de développer quelques outils nécessaires à la mise en place d'une association claims-tweet, pouvant être exploré par un utilisateur. Les seuils ont pu être utilisés pour tester un algorithme de construction de base de données, associé à la LDA permettant de vérifier si le traitement des données étaient bien similaire et les faisaient se classer dans des groupes communs. Des méthodes de filtrage, basées notamment sur les noms, ont permis de relier des éléments entre eux en partant du fait qu'elles puissent nommer des personnes similaires.

Nous n'avons pas pu pousser davantage l'étude de ces différentes méthodes, ni vérifié ce qu'il était possible de faire avec une troisième base de données. Cependant, en utilisant les méthodes de gestion de projet vues au cours du semestre, nous avons pu nous organiser et effectuer un travail de groupe sur plusieurs axes concernant ce sujet. Nous avons quand même pu dégager des résultats, et proposés différentes manières de faire des liens entre les deux bases, tout en ayant eu le désir d'aller plus loin dans nos analyses et nos résultats.

Références

- Rania Albalawi, Tet Hin Yeap, and Morad Benyoucef. Using topic modeling methods for short-text data : A comparative analysis. *Frontiers in Artificial Intelligence*, 3, 2020. ISSN 2624-8212. doi:[10.3389/frai.2020.00042](https://doi.org/10.3389/frai.2020.00042). URL <https://www.frontiersin.org/article/10.3389/frai.2020.00042>.
- Andrey. What is text vectorization ? 2021.
- Alberto Bietti. Latent dirichlet allocation. 2012. URL <https://alberto.bietti.me/files/rappor t-lda.pdf>.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null) :993–1022, mar 2003. ISSN 1532-4435.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Dimitar Dimitrov, Erdal Baran, Pavlos Fafalios, Ran Yu, Xiaofei Zhu, Matthias Zloch, and Stefan Dietze. Tweetscov19-a knowledge base of semantically annotated tweets about the covid-19 pandemic. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2991–2998, 2020.
- Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering : a brief survey. *A review of machine learning techniques for processing multimedia content*, 1 :9–16, 2004.
- Rahul Kumar Gupta, Ritu Agarwalla, Bukya Hemanth Naik, Joythish Reddy Evuri, Apil Thapa, and Thoudam Doren Singh. Prediction of research trends using lda based topic modeling. *Global Transitions Proceedings*, 2022. ISSN 2666-285X. doi:<https://doi.org/10.1016/j.gltip.2022.03.015>. URL <https://www.sciencedirect.com/science/article/pii/S2666285X22000206>.
- Alain Lelu. Filtrages et synthèses de masse sur internet. *Les cahiers du numériques*, 3 :171, 2002. URL <https://www.cairn.info/revue-les-cahiers-du-numerique-2002-1-page-171.htm>.
- Qiang Lu, Jack Conrad, Khalid Al-Kofahi, and William Keenan. Legal document clustering with built-in topic segmentation abstract. pages 383–392, 10 2011. doi:[10.1145/2063576.2063636](https://doi.org/10.1145/2063576.2063636).
- Elsa Negre. Comparaison de textes : quelques approches.... *HAL open science*, page 9, 2013a. URL <https://hal.archives-ouvertes.fr/hal-00874280/document>.
- Elsa Negre. Comparaison de textes : quelques approches... 2013b.
- Nils Reimers and Iryna Gurevych. Sentence-bert : Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
- Jonas Rieger, Jörg Rahnenführer, and Carsten Jentsch. Improving latent dirichlet allocation : On reliability of the novel method ldaprotoype. In Elisabeth Métais, Farid Meziane, Helmut Horacek, and Philipp Cimiano, editors, *Natural Language Processing and Information Systems*, pages 118–125, Cham, 2020. Springer International Publishing. ISBN 978-3-030-51310-8.
- Catarina Silva and Bernardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks*, 2003., volume 3, pages 1661–1666. IEEE, 2003.
- Christoph Stanik, Tim Pietz, and Walid Maalej. Unsupervised topic discovery in user comments, 2021. URL <https://arxiv.org/abs/2108.08543>.

Sandeep Tata and Jignesh M Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, 36(2) :7–12, 2007.

Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. Claimskg : a knowledge graph of fact-checked claims. In *International Semantic Web Conference*, pages 309–324. Springer, 2019.

Juntao Wang and Xiaolong Su. An improved k-means clustering algorithm. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 44–46, 2011. doi:[10.1109/ICCSN.2011.6014384](https://doi.org/10.1109/ICCSN.2011.6014384).

A Annexes

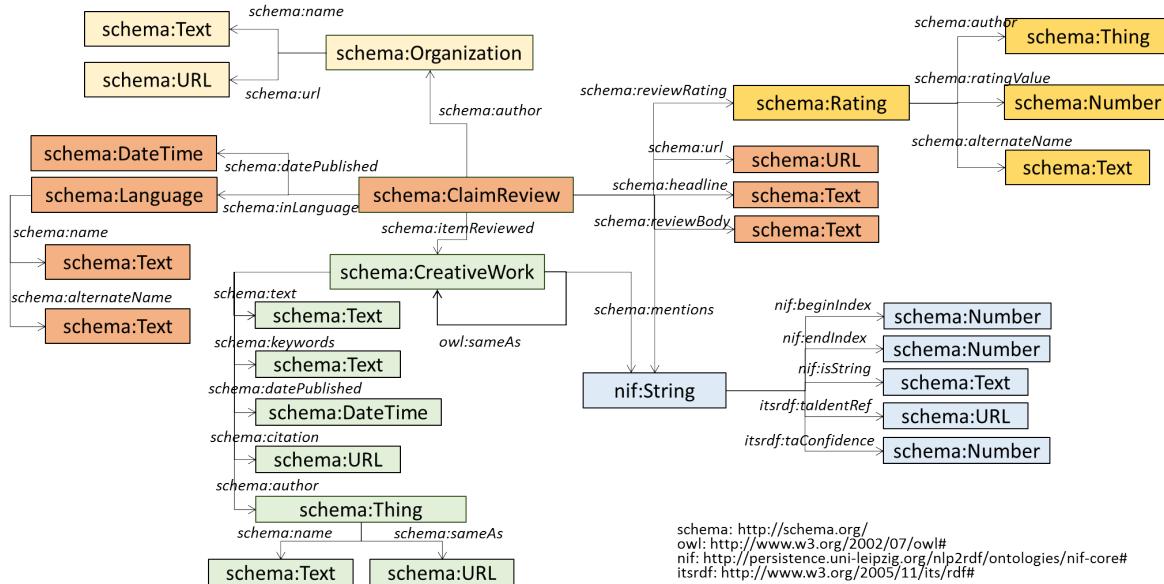


FIGURE 11 – Le modèle de données de la base ClaimsKG

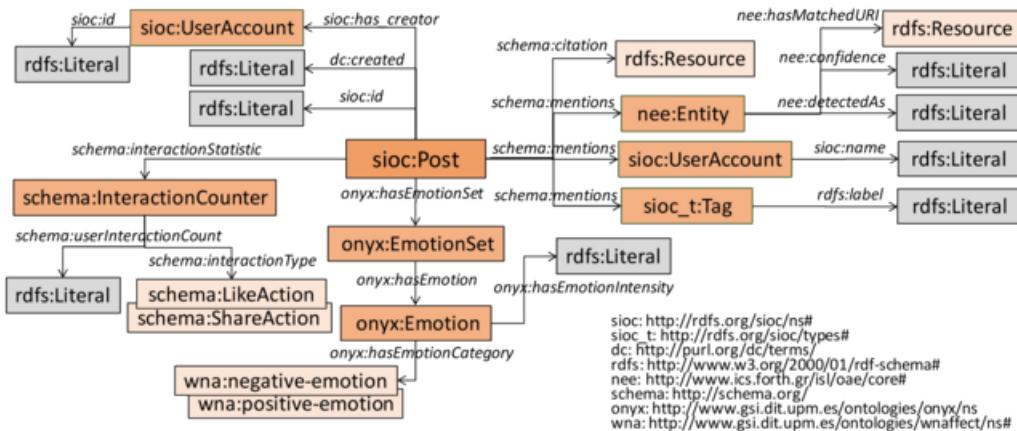


FIGURE 12 – Le modèle de données de la base TweetsCOV19

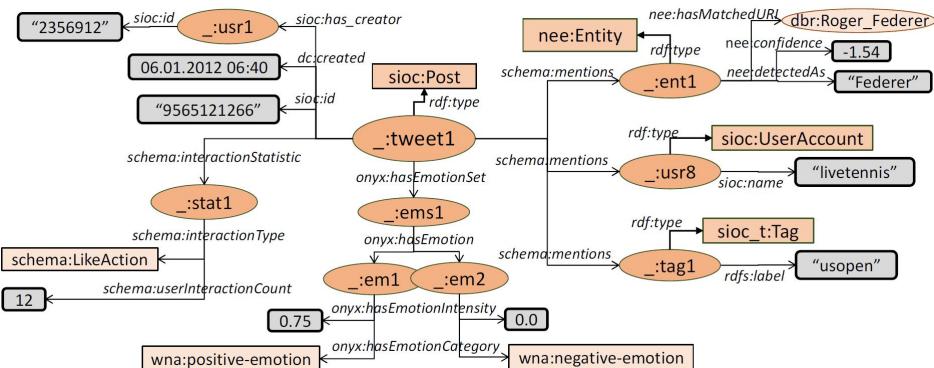


FIGURE 13 – Un exemple de version instanciée du modèle de données de la base TweetsCOV19

Hierarchy of annotations: Check in order of appearance and fallback on next if necessary: E -> E* -> ST -> N -> O Where E = Exact Match; E* = Same claim by != people; ST = Same Topic; N = No Match; O = Other Use only the information available in the titles, links entities, if you need to check the sources then be conservative and fall back on the next applicable class that is certain to be true without external inference							
Annotations	Score	CR Author , CR Author E Review URL A	Review URL B	Text Fragments A	Text Fragments B	Entities A	Entities B

FIGURE 14 – Annotations et attributs de Claims2

```

Label : 6 taille : 55 Éléments du premier document : 38 , Éléments du deuxième document : 17
160 Did Scott Walker flip-flop on how budget short...
288 Did Scott Walker flip-flop on a immigration an...
394 Did Scott Walker flip-flop on a immigration an...
403 Did Scott Walker flip-flop on how budget short...
898 Scott Walker says 2013 job growth in Wisconsin...
dtype: object

Label : 7 taille : 52 Éléments du premier document : 22 , Éléments du deuxième document : 30
865 U.S. Senate candidate Barry Hinckley says Sen....
872 Has PolitiFact rated Scott Walker the nation's...
1072 Has PolitiFact rated Scott Walker the nation's...
1101 U.S. Senate candidate Barry Hinckley says incu...
1158 U.S. Senate candidate Barry Hinckley says Sen....
dtype: object

Label : 8 taille : 32 Éléments du premier document : 16 , Éléments du deuxième document : 16
266 Republican Bernard Jackvony says Democrat Davi...
434 Republican Bernard Jackvony says Democrat Davi...
720 Brendan Doherty says U.S. Rep. David Cicilline...
922 Talk show host John DePetro says Providence wa...
1133 Brendan Doherty says U.S. Rep. David Cicilline...
dtype: object

Label : 9 taille : 89 Éléments du premier document : 54 , Éléments du deuxième document : 35
81 Rick Santorum has it wrong on Mitt Romney's book
344 Mitt Romney says Barack Obama 'could have gott...
371 Mitt Romney's views on climate change have cha...
440 Rick Perry says Barack Obama is a socialist
964 Mitt Romney rejected state tax pledge before s...
dtype: object

```

FIGURE 15 – Visualisation de la composition de 4 sujets sur 10 sortis par KMeans

```
[0,
 '0.061*"year" + 0.023*"rate" + 0.016*"number" + 0.013*"state" + '
 '0.013*"countri" + 0.010*"percent" + 0.010*"school" + 0.008*"increa" + '
 '0.008*"world" + 0.008*"home"),
(1,
 '0.015*"today" + 0.015*"citizenship" + 0.009*"year" + 0.009*"presid" + '
 '0.009*"cost" + 0.009*"child" + 0.009*"want" + 0.009*"vote" + 0.009*"peopl" +
 '+ 0.009*"black"),
(2,
 '0.021*"govern" + 0.018*"death" + 0.016*"rate" + 0.013*"pay" + '
 '0.011*"actual" + 0.011*"system" + 0.011*"right" + 0.011*"girl" +
 '0.011*"read" + 0.008*"state"),
(3,
 '0.038*"presid" + 0.023*"trump" + 0.013*"time" + 0.013*"govern" +
 '0.011*"state" + 0.011*"barack" + 0.009*"tax" + 0.009*"cut" + 0.009*"vaccin" +
 '+ 0.009*"covid"),
(4,
 '0.016*"money" + 0.016*"pay" + 0.013*"state" + 0.013*"oil" + 0.010*"percent" +
 '+ 0.010*"campaign" + 0.010*"term" + 0.010*"month" + 0.007*"govern" +
 '0.007*"presid"),
(5,
 '0.017*"parti" + 0.014*"presid" + 0.014*"govern" + 0.010*"state" +
 '0.010*"barack" + 0.007*"year" + 0.007*"school" + 0.007*"rate" +
 '0.007*"even" + 0.007*"vaccin"),
(6,
 '0.020*"tax" + 0.017*"happen" + 0.011*"beer" + 0.011*"even" + 0.011*"think" +
 '+ 0.011*"water" + 0.009*"health" + 0.009*"care" + 0.009*"percent" +
 '0.009*"today"),
(7,
 '0.031*"percent" + 0.026*"hous" + 0.017*"tax" + 0.015*"time" +
 '0.012*"govern" + 0.012*"spend" + 0.010*"year" + 0.010*"state" +
 '0.010*"fire" + 0.010*"plan"),

```

FIGURE 16 – Visualisation de la composition de 8 sujets sur 10 sortis par la LDA

```
Label : 4 taille : 65 Éléments du premier document : 25 , Éléments du deuxième document : 40
211 methotrexate rpr nitrous oxide cytology copper...
605 opposition net neutrality repeal
662 nevada recently unemployment crime bankruptcy ...
687 never anyone senate
925 continually try wrap withdrawal agreement unio...
dtype: object

Label : 5 taille : 62 Éléments du premier document : 27 , Éléments du deuxième document : 35
622 video propaganda report today
908 number benefit estate tax break
922 president barack president keep dog trainer re...
926 support approval rating morning party time record
963 theresa drop deal implementation period end tr...
dtype: object

Label : 6 taille : 37 Éléments du premier document : 14 , Éléments du deuxième document : 23
14 filling station ashaiman see you tomorrow
216 snap election brussels glasgow brexit moscow s...
491 opioid head on straight tx shasta
531 video spider inside burst peel power
741 workforce number fuel industry
dtype: object

Label : 7 taille : 39 Éléments du premier document : 10 , Éléments du deuxième document : 29
551 message line include virus
585 gang city rob kill random stop
737 century man edward mordrake also condition for...
789 photographs show fighter jet air
921 year increase
dtype: object

Label : 8 taille : 70 Éléments du premier document : 23 , Éléments du deuxième document : 47
496 my bff it s a miracle seatbelt they re alive t...
583 year boy fbi
764 intelligence hurriedly program language
831 president ceo godfather pizza go sense business
917 driver drove wall look tunnel manner looney ro...
```

FIGURE 17 – Composition de sujets obtenus après LDA, claims vs tweets

The screenshot shows a Trello board titled "Objectifs du projet". Under the "Description" section, it says "Les principales tâches que l'on doit effectuer pour le TER". A checklist titled "Chronogramme général" is displayed, with a progress bar at 80% completion. The tasks listed are:

- 8-Mars—Analyse et exploitation des données
- 12-Avril—Match Tweet—Claim
- 3-Mai—Regroupement par Thèmes
- 17 Mai - Lien avec la troisième base
- 23-Mai—Rendu du Projet

Buttons for "Add an item", "Hide checked items", and "Delete" are visible.

FIGURE 18 – Feuille de route du projet, sur Trello