Rapport de projet entreprise

NLP et Rédaction de demande de brevets

Réalisé par
Adrien COLMAGRO
Maxime GOIN
Nicolas TIREL
Thomas VENANT-VALERY

ING3 - 2020/2021

SOMMAIRE

1. CLASSIFICATION DES DEMANDES DE BREVETS

1.1. Données

A partir des données fournies par l'entreprise pour effectuer la prédiction d'une classification d'un brevet, nous avons décidé de trier et de récupérer seulement les données qui nous semblaient les plus pertinentes pour atteindre nos objectifs. Nous avons d'abord décidé d'étudier les données et de visualiser la répartition et le type de données que nous avions à disposition. Ainsi, nous avons remarqué la présence de trois langues différentes à traiter, chaque brevet possédant son titre, son abrégé, sa description et ses revendications dans une langue particulière entre le français, l'anglais et l'allemand.

Les modèles de classification appliqués à des données textuelles ne pouvant fonctionner que dans une seule langue à la fois, nous avons rapidement pris la décision de séparer les trois langues en des jeux de données distincts. Faire ainsi nous a permis de répartir les tâches, et de disposer pour la suite d'une base plus adaptée pour poursuivre en fonction de la langue à traiter.

A partir de ce premier traitement, nous avons ensuite retirer toutes les données qui ne semblaient pas pertinentes pour déterminer une classification : tout d'abord le type de classification, qui exprimait le degré de certification d'un brevet (publié par un rapport de recherche européen ou non, validé par une gratification ou une nouvelle spécification…). Par exemple, certains brevets existaient en double dans le dataset avec deux types de classification différente, nous n'avons gardé que la version plus pertinente dans ce cas là, puisqu'il s'agit d'une correction du brevet précédent.

Nous avons ensuite enlevé des informations relatives à la classification comme la date de la classification ainsi que la date de publication : nos données datant toutes des années 2019 à aujourd'hui, il nous a paru logique qu'une différence de quelques jours ou mois ne pouvait expliquer une classification différente. A partir de là, il nous restait pour un brevet son numéro, qui pouvait servir à l'identification, ses revendications, sa description, son titre, son abrégé et finalement sa classification, la donnée que l'on cherche à prédire.

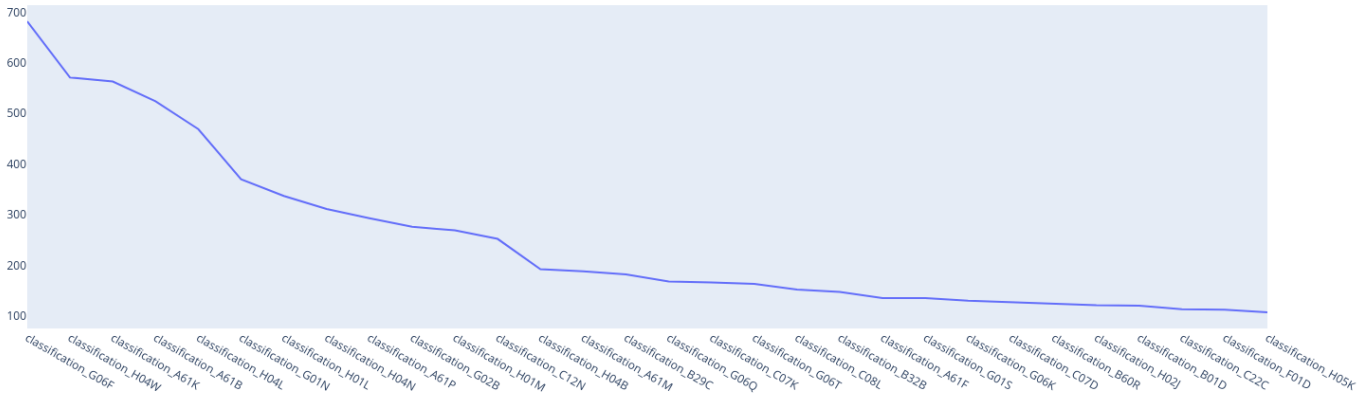
Une fois ce premier ménage accompli, nous avons dû effectuer un certain nombre de traitements supplémentaires avant de disposer de données utilisables à l'entraînement de notre modèle. L'élément le plus simple à traiter est le titre, que nous avons normalisé en lettres minuscules pour se défaire des typographies exotiques.

Pour les autres parties (abrégé, description et revendications), il a été nécessaire de pousser la réflexion bien plus loin pour chasser les composants superflus, ainsi que les composants néfastes au processus d'apprentissage. Nous avons supposé par exemple qu'il n'était pas judicieux de conserver les titres des sections dans la description, ou les descriptions de figures. Toutes ces données sont organisées sous la forme d'un document xml, donc nous avons pu accéder directement aux tags intéressants et surtout localiser ceux à retirer. Une fois le triage effectué, toutes les balises furent supprimées pour garder uniquement le texte concernant directement le brevet en question. Le texte résultant, bien que lisible, n'est pas tout à fait adapté pour poursuivre.

Les dernières étapes du traitement nécessitent à ce stade de s'occuper des aspects syntaxiques et sémantiques. La ponctuation par exemple, est comparable à de la décoration et ne fournit pas d'information supplémentaire. Les nombres, bien qu'indispensables dans les explications relatives à la propriété intellectuelle, ne serviront eux non plus à établir une classification de meilleure qualité. Il existe aussi des listes de mots qui sont si fréquents, qu'il est inutile de s'en servir pour chercher de l'information, ce sont les mots vides, ou stop words en anglais (exemples: "the", "a", "and", "or"…).

Toutes ces mesures de filtrage permettent d'éliminer une partie non-négligeable de la masse à traiter disposant de peu de valeur exploitable. Enfin les éléments survivants seront par la suite tokenisés, action consistant à transformer un texte en un vecteur de nombres, exploitable dans un modèle de prédiction.

In fine, toutes ces étapes permettent de réduire le traitement et d'améliorer la performance des modèles en lui fournissant uniquement les données les plus pertinentes.



La répartition des 30 sous-classes les plus présentes dans nos données

1.2. Modèles

Deux modèles ont été envisagés pour prédire la classification d'un brevet en anglais, et un autre modèle a été utilisé pour prédire la classification d'un brevet en français.

Pour le premier modèle anglais, nous avons choisi d'utiliser un réseau de neurones de plusieurs couches, une couche d'entrée, une couche d'embedding, une couche LSTM et finalement une couche dense pour obtenir la prédiction. Ce réseau prend en entrée l'un des texte parmi la description, le titre, l'abrégé ou les revendications, et va la tokenizer puis utiliser un embedding à l'aide du document glove.6B.100d.txt, qui vont permettre de connecter des mots qui se rapprochent dans leurs racines ou leurs significations.

Ce modèle a été testé à partir de différents hyper paramètres, puis en modifiant le texte en entrée, mais finalement n'a pas été retenu car il ne permettait pas de classer les brevets par sous-classes avec une précision supérieure à 5%.

Nous sommes ensuite passés sur un modèle appliquant BERT, et utilisant les transformers proposés par Google. Tout comme le modèle précédent, nous avons pu tester plusieurs configurations et trouver la plus performante pour convenir à nos besoins.

Le choix des modèles s'explique avant tout par leurs performances respectives, mais aussi par la compréhension de leur utilisation. Pour le premier, nous nous sommes bien rendus compte que son usage était limité vis-à-vis de la tâche à effectuer. Il était nécessaire pour nous d'avoir un modèle capable de classer les brevets selon un nombre important de sous-classes possibles. Il s'agit d'un cas de prédiction multi-classe, c'est-à-dire la prédiction à effectuer peut concerner 3 catégories différentes ou plus. De plus, la classification est multi-label, ce qui signifie qu'un brevet peut appartenir à plusieurs classes en simultané. Ainsi la classification ne peut se contenter uniquement de la classe la plus vraisemblable.

L'emploi de BERT répondait alors davantage à notre problème, car ce modèle est capable de s'entraîner sur une grande capacité de données, et de garder le plus fidèlement possible la signification de ces données. Cela nous a donc permis d'obtenir une précision sur la classification de certains brevets qui apparaissent le plus souvent de plus de 90% !

```
F1 Score (Weighted) = 0.6509621910405511
F1 Score (Samples) = 0.654620401132029
F1 Score (classification_H04W) = 0.7314974182444062
F1 Score (classification_H04L) = 0.6425255338904363
F1 Score (classification_H01L) = 0.8083209509658246
F1 Score (classification_H04N) = 0.8086642599277978
F1 Score (classification_H01M) = 0.934131736526946
F1 Score (classification_H04B) = 0.4736842105263158
F1 Score (classification_H02J) = 0.5617977528089888
F1 Score (classification_H05K) = 0.32116788321167883
F1 Score (classification_H04M) = 0.4484848484848485
F1 Score (classification_H05B) = 0.6569343065693432
F1 Score (classification_H02K) = 0.7724137931034483
F1 Score (classification_H02M) = 0.6440677966101694
F1 Score (classification_H01R) = 0.7175572519083969
F1 Score (classification_H04R) = 0.8148148148148148
F1 Score (classification_H01Q) = 0.7526881720430108
F1 Score (classification_H01F) = 0.34375000000000006
F1 Score (classification_H02P) = 0.2857142857142857
```

Score F1 de classification par sous-classes de brevets avec l'emploi de BERT

1.3. Modèle alternatif

En ce qui concerne le modèle pour classifier les brevets français, nous avons commencé par utiliser CamemBERT grâce à la librairie Hugging Face, version dérivée de BERT et adaptée à la langue française, développée par l'INRA.

Cependant les résultats se sont rapidement avérés loin des attentes, et inférieurs à ceux obtenus sur les brevets anglais. Le problème provient de la large différence dans les quantités respectives de données utilisables. Suite à ce constat, la nécessité de se défaire du manque des données nous a poussé à explorer une façon différente d'aborder le problème.

Ce nouvel angle d'attaque consiste à utiliser la définition officielle de chaque catégorie, pour la comparer avec le brevet, dans le but de déterminer si le brevet correspond à ladite catégorie.

Nous avons récupéré des documentations au format XML, sur le site wipo.int concernant le système de classification IPC en vigueur, contenant notamment l'architecture complète de ce dernier, et la définition de plusieurs dizaines de milliers de catégories distinctes.

Avec la librairie *AnyTree*, nous avons reconstitué en python l'arborescence complète de classification IPC, dans laquelle le premier niveau correspond aux Sections et à leur description, le deuxième aux Classes, puis aux Sous-Classes, etc. Le schéma complet s'apparente à un arbre de décision, dans lequel on suit les branches, en augmentant à chaque étage le niveau de détail dans la classification.

Le traitement pour la création de l'arbre fut fastidieux, beaucoup d'éléments parasites dans notre cas on du être filtrés, tels que les notes, les références à d'autres catégories ou les regroupements. Nous avons également appliqué des traitements aux descriptions obtenues afin de réduire au mieux les informations non essentielles (ponctuation, stop words).

L'avantage d'avoir un tel arbre réside dans le nombre d'opérations à effectuer. Pour classifier un brevet, la première étape est de le comparer aux définitions accessibles au premier niveau de l'arbre, soit un nombre très réduit, 8 dans notre cas. Uniquement certaines vont produire un résultat positif, la recherche se poursuivra pour celles-ci au deuxième niveau, tandis qu'elle sera interrompue pour les autres.

Cette idée bien que prometteuse n'a jamais porté ces fruits. Plusieurs obstacles n'ont pas été surmontés, tel que le critère de décision lors du parcours de l'arbre. Certains critères atteignaient pour un brevet l'extrémité de l'arbre sur plusieurs milliers de classifications, tandis que d'autres ne dépassaient pas le cap des Sous-Classes. Un autre problème similaire était de savoir si un critère n'était plus valable passé une certaine profondeur d'exploration.

A cela il faut ajouter que la structure initiale de l'arbre, telle que défini par l'IPC n'est sans doute pas optimale. Les premiers niveaux ont des descriptions extrêmement

générales. Comment expliciter qu'un brevet concernant un barbecue doit passer le test "nécessités de la vie courante" ?

Deux pistes d'améliorations sont à envisager pour mener à bien la preuve de concept. Réorganiser la structure de l'arbre de décision, pour échapper aux définitions trop générales, par exemple en ayant au premier niveau, les Classes, ou les Sous-Classes, ou même une agrégation de définitions des 3 premiers niveaux initiaux. Une autre piste à explorer consiste à un critère qui évolue en fonction de la profondeur d'exploration, pour être mieux calibré.

1.4. Entraînement

Pour les entraînements, nous avons décidé de séparer nos données, à hauteur de 80% dédié aux phases d'entraînement des modèles, et 20% pour évaluer et contrôler le déroulement des entraînements.

Lors des phases d'entraînement, le modèle est malléable et cherche à s'ajuster aux données qu'il utilise. Il a connaissance des prédictions erronées qu'il a produites et s'en sert pour converger vers un état propice à réussir un maximum de classifications. Ainsi on dit qu'il apprend. On appelle une époque, le passage de toutes les données d'apprentissage par le modèle.

A la fin de chaque époque, l'état du modèle est figé, on lui soumet les 20% des données qu'il n'a jamais rencontrées, afin d'évaluer sa capacité à généraliser et donc à être exploitable. On compare le résultat de l'évaluation avec ceux des entraînements pour déceler des irrégularités éventuelles, et les corriger en jouant sur les paramètres à disposition lors de la phase d'apprentissage. On vérifie également la bonne évolution du processus entre les époques, il est généralement nécessaire d'utiliser à plusieurs reprises l'échantillon d'entraînement sur le même modèle.

Notre objectif est de s'assurer que la précision de la classification en situation d'évaluation ne cesse de grimper, et que la quantité d'erreurs diminue continuellement.

Si l'on constate que le modèle s'améliore encore lors des entraînements, mais régresse sur les évaluations, c'est le signe qu'il commence à connaître par coeur les données d'apprentissage plutôt que de généraliser ce qu'il a appris, on appelle cela le surapprentissage (ou overfitting en anglais).

Quand cela nous est arrivé, nous avons relancé l'entraînement en diminuant le nombre d'époques afin d'interrompre le processus avant d'atteindre ce palier de non-retour.

1.5. Résultat et Optimisation

Pour évaluer la pertinence du modèle, nous évaluons le score F1 et la précision moyenne du modèle sur les données de test une fois que le modèle a fini de s'entraîner, ainsi que le score F1 de chaque sous-classe à prédire. Comme il a été dit auparavant, une fois un score obtenu, on teste alors de nouveau le modèle en l'entraînant avec de nouveaux hyper paramètres et on compare si les résultats nous semblent meilleurs ou pas.

À l'aide du modèle BERT, nous avons constaté que la classification de certaines sous-classe se bloquait à 0% de précision. En regardant les brevets disponibles en entraînement et correspondant à ladite classification, nous avons remarqué qu'il s'agissait toujours de catégories pour lesquelles on possède un échantillon d'exemples très restreint. Par conséquent, les classifications correspondantes en deviennent marginales, et le modèle ne s'adapte pas assez pour en tenir compte.

De plus, lorsque ces catégories échappent au zéro pointé, c'est au mieux pour approcher de 10% maximum, lorsqu'on dispose d'un peu plus de données sur ces cas. Nous avons donc choisi de garder uniquement les sous-classes qui contiennent un seuil minimum de brevets et ainsi assurer une certaine qualité sur la prédiction. Ce choix nous semble le plus approprié car le rééquilibrage du jeu de données est hors de portée.

En comparant nos résultats avec l'état de l'art antérieur, on constate que notre précision est légèrement en dessous de ce qu'ont pu proposer les articles scientifiques sur le sujet. Nos résultats les plus convaincants ont été obtenus en utilisant les titres, les abrégés et les descriptions des brevets, au détriment des revendications. Nos meilleurs modèles atteignant environ 70% de prédiction correctes contre plus de 90% dans leur recherches.

Cela peut s'expliquer par le manque de données que nous possédons, et par la complexité de la tâche demandée. Effectivement, il est d'usage de prédire entre 5 à 10 classes dans les problèmes classiques de NLP, à l'aide de plusieurs dizaines voir centaines de milliers de représentants par catégorie.

De notre côté, nous avons un total d'environ 40.000 brevets anglais, ou 10.000 en français à classer dans 500 sous-classes possibles, ce qui ne permet pas d'atteindre un quantité convenable d'exemples par sous-classes. De plus, la disparité dans la répartition des brevets est flagrante, car de nombreuses sous-classes disposent d'une dizaine de représentants seulement, tandis que d'autres en possèdent des milliers. Cela justifie d'autant plus notre choix de réduire les données d'entraînement avec les sous-classes qui apparaissent peu souvent, pour diminuer significativement le nombre de sous-classes à prédire.

Du côté des recherches scientifiques, ils ont à leur disposition un ensemble supérieur à 2 millions de brevets, fournissant plus de grain à moudre concernant les classes sous-représentées. Avec de telles quantités de données, les possibilités de performances sont grandement améliorées, et ces équipes peuvent ainsi prétendre à l'obtention d'une bien meilleure précision que nos modèles.

2. GÉNÉRATION DE LA DESCRIPTION OU D'UNE PARTIE DE LA DESCRIPTION

2.1. Données

En ce qui concerne les données utilisées pour la partie génération, nous avons pu réutiliser en grande partie le travail de préparation des données fait lors de la classification. Pour générer une partie de la description en utilisant une revendication, nous avons récupéré pour chaque langue, les revendications et descriptions. Les descriptions étant en général très longues, cela augmente grandement la taille des variables utilisées pour stocker les données. Comme ces données textuelles contenaient des balises XML, il a fallu dans un premier temps les supprimer. Pour cela, nous avons utilisé la librairie BeautifulSoup. Ainsi, nous pouvions désormais utiliser ce texte brut.

Ensuite, on a créé une fonction qui nettoie le texte. Cette dernière passe le texte en minuscule, supprime les parenthèses et leur contenu, car cela réfère souvent à des points définis ailleurs dans le brevet qui seraient inutilisables dans le modèle et supprime des caractères spéciaux. En appliquant cette fonction aux revendications et descriptions, on obtient alors un nouveau dataframe avec le texte tel que voulu pour le modèle. Selon les langues, certains ajustements peuvent être faits, par exemple le format des guillemets : “ ” (anglais), « » (français), » « (allemand).

À cause de la taille en mémoire des descriptions et des limites matérielles que nous avons, on a fait le choix pour éviter les crashs et entraînements interminables de ne garder qu'un pourcent des descriptions. En faisant cela, notre modèle va inévitablement perdre en précision. Toutefois, il suffirait de supprimer ces quelques lignes de code si l'on avait accès à des machines plus puissantes.

On a ensuite construit deux dictionnaires qui vont coder les différents caractères uniques trouvés dans la description ; une dans le sens caractère -> indice et l'autre dans le sens indice -> caractère. Cela étant fait, on a aussi créé une liste de séquences en utilisant 40 caractères par séquence (plus poserait problème à cause de la mémoire limitée dont on dispose). On générera ainsi une partie de la description, mais ce procédé est réitérable afin d'en générer plus au besoin. Enfin, on a transformé nos données textuelles en matrices numpy x et y, afin que leur format soit adapté à l'entraînement du modèle.

2.2. Modèles

En ce qui concerne le choix du modèle, nous en avons envisagé deux différents. Le premier est le modèle GPT-2. Nous avons donc d'abord essayé de mettre en place celui-ci, mais sans succès. En parallèle, nous avons essayé un second modèle qui est le modèle LSTM. C'est un modèle que l'on peut avoir avec la librairie keras, il ressemble donc plus que GPT-2 aux différents modèles que nous avons vus cette année. Néanmoins, d'après la documentation que l'on a pu trouver en ligne, le modèle GPT-2 semble être souvent privilégié dans les tâches de génération de texte. Nous supposons donc que notre modèle n'obtiendra pas de résultats aussi bons que ceux que l'on aurait pu avoir avec GPT-2. Toutefois, notre modèle étant plus proche de ce que l'on connaît, il est plus facile à comprendre et à modifier.

On utilise dans le modèle deux couches LSTM avec 128 neurones chacune. On pourrait envisager comme piste d'amélioration d'ajouter des couches et des neurones. On ajoute aussi une couche Dense et l'activation en softmax. On utilise aussi un optimizer RMSprop avec un coefficient de 0,01. Ce coefficient a un certain impact sur le résultat final et cette valeur semble être assez adaptée à ce que l'on recherche.

On a aussi défini deux fonctions qui serviront lors de l'entraînement. La première échantillonne un index à partir d'un tableau de probabilités et du paramètre de température. Ce paramètre définit la "créativité" lors de la génération : plus il est élevé, plus le texte généré sera créatif. La seconde fonction va afficher le texte généré pour certaines epochs pour différentes valeurs de températures.

2.3. Entraînement

Pour l'entraînement, on a dû le limiter afin d'avoir des temps d'entraînement raisonnables. Comme évoqué plus haut, on utilise qu'un pourcent des descriptions à cause de la grande taille mémoire qu'elles occupent. On utilise une batch_size de 128 et 1 epoch. Nous avons essayé d'aller jusqu'à 10 epochs, mais cela posait deux problèmes : le premier étant le temps très long d'entraînement (entre 500 et 1000 secondes par epoch) et le second étant l'overfitting. On a pu constater de l'overfitting dès la seconde epoch ; d'où la limite à 1 epoch.

Suite à l'entraînement, on affiche des textes générés à partir d'un morceau de phrase. L'algorithme va ensuite essayer de la compléter dans la limite de caractères que l'on a définis plus tôt. On génère des phrases pour des valeurs de températures de 0.2, 0.3, 0.4 et 0.5 car ce sont les valeurs qui produisent les résultats les plus satisfaisants.

Nous n'avons pas segmenté les données en test/train, car il n'est pas pertinent selon nous de chercher à comparer le texte obtenu avec une description entière. On justifie cela par la différence de taille entre le texte généré et la description.

2.4. Optimisation

Comme évoqué précédemment, il n'existe pas vraiment de metrics capable de donner un bon avis sur la qualité du texte généré. Pour évaluer notre modèle, on a donc regardé quelques phrases afin de constater si les phrases faisaient sens. On a pu en tirer plusieurs conclusions.

D'abord, on constate que le texte généré contient des mots qui existent pour la plupart, mais n'écrit jamais de point. Cela fait donc des phrases très longues qui vont tendre à perdre en sens au fur et à mesure. Un aspect positif est que lorsque la phrase en entrée se termine au milieu d'un mot, en raison de la limite de caractères (ex: "le chat est pe"), l'algorithme va souvent compléter le dernier mot afin qu'il ait un sens ; dans cet exemple il complétera "pe" en "petit".

Nous avons plusieurs pistes d'améliorations à ce sujet. La première étant d'utiliser des machines plus puissantes pour pouvoir utiliser plus de données en mémoire vive. Cela améliorera les modèles. Une seconde piste serait d'essayer à nouveau d'implémenter GPT-2, on pourrait alors comparer les deux modèles.

Cependant, il est peu probable d'atteindre un texte généré très proche d'une véritable description. C'est dû notamment à la grande diversité des sujets que l'on peut trouver dans les données et la complexité/rareté de certains termes utilisés.

Pour aller plus loin, on a réfléchi à la meilleure façon de mettre en place la génération de description. Comme notre modèle ne semble pas pouvoir générer de longs textes tout en conservant la cohérence de ces derniers, on pense qu'un bon moyen de générer une description serait de trouver les longueurs moyennes des phrases du paragraphe de revendications et celle des phrases de la description.

Ainsi, on générerait des phrases ayant comme longueur maximale la longueur moyenne d'une phrase de description. On pourrait donner à l'algorithme comme base les premières moitié des phrases de revendications, ainsi, il pourrait les compléter, mais pour la description. On aurait alors une description générée à partir des revendications. En se limitant à ces demies-phrases, la description générée serait meilleure, car on générerait des phrases plus courtes ce qui rendrait l'apprentissage et la restitution meilleurs.

3. GÉNÉRATION D'UN ABRÉGÉ À PARTIR DES REVENDICATIONS

Pour la génération d'un abrégé à partir de revendications, nous avons pu utiliser une bonne partie du travail déjà réalisé lors de la partie précédente. On commence par récupérer les données qui nous intéressent. Puis on applique un premier traitement pour nettoyer les textes des résumés et des revendications (même traitement que pour la génération de descriptions). On récupère aussi les ''stopwords'' selon la langue que l'on traite.

On cherche ensuite à construire une liste des phrases ayant le plus d'informations importantes des revendications. La méthode que l'on a adoptée pour cette partie ne repose pas sur de l'IA, mais sur des notions de NLP seulement. On va donc chercher à comparer par la suite la phrase d'un abrégé avec la phrase ayant le plus de sens dans un paragraphe de revendications.

On détermine la phrase ayant le plus de sens après avoir créé un dictionnaire des fréquences des mots d'une revendication tout en ne prenant pas en compte les stopwords. Ainsi, on va pouvoir donner un poids aux mots afin de déterminer les plus importants. En effet, on a constaté que dans les revendications, les mots importants sont souvent utilisés et répétés. Ensuite, on va pouvoir donner un score à chaque phrase d'un texte de revendication en fonction des mots qui la composent. On récupère alors celle avec le score le plus élevé.

Comme notre méthode n'utilise pas l'IA, il fallait que l'on puisse l'évaluer. On a donc retenu deux méthodes qui peuvent comparer deux phrases. La première utilise la fonction SequenceMatcher de la librairie difflib. Le résultat est assez moyen en utilisant cette méthode. On a donc développé une fonction de comparaison un peu plus poussée. D'abord, on retire les stopwords des deux phrases. On construit alors deux listes qui contiennent des 0 et des 1 pour indiquer si la phrase contient ou non le mot en n-ième position. Enfin, on utilise la formule "cosine". Cela consiste à diviser le nombre de mots en commun par la racine carrée de la somme du nombre de mots des deux listes. On obtient alors une valeur entre 0 et 1 qui estime la similarité entre les deux phrases. Avec cette méthode, on a des résultats presque deux fois meilleurs qu'avec la méthode précédente.

	Français	Anglais	Allemand
Méthode naïve	35.0%	20.3%	38.0%
Méthode avancée	66.8%	54.3%	60.0%

On constate que les résultats sont alors assez satisfaisants. Notre méthode est donc adaptée pour générer un abrégé à partir d'une revendication. On garde à l'esprit qu'elle ne génère pas à proprement parler un nouvel abrégé, mais on obtient au final un résultat qui pourrait être meilleur ou au moins équivalent à un texte généré par une IA.

Pour aller plus loin, on pourrait envisager un pré-traitement plus pointu selon les langues et aussi une comparaison de notre méthode avec une méthode utilisant l'intelligence artificielle.