

Système de traitement de textes Hubbix

Projet de fin d'année
Développeur en Intelligence Artificielle

2021 / 2022



MURO R. Daniel Alexander

Table des matières

Table des matières	2
Remerciements	3
NLP pour Hubbix : notre besoin	4
La data	5
Nos données	5
Nos sources	5
Le prétraitement et la sauvegarde	7
L'Intelligence artificielle	7
La préparation des données	7
Après avoir récupéré le document et sauvegardé ses articles, l'utilisateur aura la possibilité de démarrer le traitement NLP des données. Dans le cadre d'un document législatif, et vu que l'objectif final de notre système d'IA est de les classifier les articles légaux selon son domaine d'application et ne pas selon son sentiment général, il sera nécessaire de garder tous les mots, sans enlever même celles considérées vides, communément appelées "stop words", celles qui n'apparaissent que peu des fois, ou celles qui n'ont pas une longueur minimale prédéfinie. Tous les mots devraient être conservés intégralement. Nous n'aurons pas non plus à lemmatiser notre texte. Les seuls pré-traitements à faire seront la suppression des signes de ponctuation et des espaces vides multiples, ainsi que la séparation de mots composées et unies par un tiret.	7
Apprentissage supervisé vs. non supervisé	7
La représentation du texte	8
Le modèle	10
Les technologies	11

Remerciements

Je tiens à remercier en premier lieu l'équipe pédagogique de la 3W Academy, qui nous a accompagnés et nous a menés à terme de cette année de formation. Je souhaite remercier très spécialement LUCSKO Antoine, SAVART Aurélien, IMLYHEN Yasmina et CARRIERE Maxime.

Je remercie également à l'ensemble de mes supérieurs chez EBP Informatique : M. GARDETTE Pierre, Responsable d'Équipe SI, M. CAHUZAC Jacques, Chef de Projet Hubbix Paie et mon tuteur et chef direct, et M. HOLTZMAN Eric, Directeur du Pôle Développement, pour sa disponibilité, pour être toujours à l'écoute, pour son attention constante et sa confiance tout au long de cette année d'apprentissage.

M. ALLAOUI Khalid, développeur Frontend, pour toutes les connaissances apportées à propos de l'utilisation d'Angular dans le cadre de ce projet.

M. DEGROUX Mathieu, Expert Technique, pour son accompagnement et pour sa disposition à toujours partager ses connaissances.

EN dernier, mais pas moins importants pour cela, je tiens aussi à remercier l'ensemble des membres du DEV SI et de l'équipe Hubbix Paie, pour leur participation à ma formation, et pour les conseils qu'ils ont su me donner au quotidien.

NLP pour Hubbix : notre besoin

EBP Informatique est bien connue pour développer et proposer une grande variété des logiciels de gestion et comptabilité, notamment ceux pour le calcul et traitement de la paie des salariés.

Dans l'univers des logiciels de gestion de paie, il y a énormément de considérations législatives à prendre en considération, et le paramétrage des règles qui affectent le calcul des cotisations, montants à payer, congés et autres valeurs, est une des étapes les plus longues et délicates. Ce paramétrage est fait par des consultants de paie et, pour cela, il est nécessaire, d'abord, de lire et interpréter une grande quantité de textes légaux qui peuvent contenir des informations liées aux règles de paie, ou même des conditions qui vont affecter une ou plusieurs règles déjà définies, ou vont rajouter des nouvelles règles au système.

Une fois que chaque document légal a été lu en sa totalité, il faut identifier quels sont les articles qui vont vraiment affecter la procédure de création et paramétrage des règles. Ensuite, il est nécessaire d'interpréter ce texte et le transformer en formule de calcul pour, finalement, pouvoir saisir cette dernière dans le logiciel afin de permettre le calcul des salaires, coûts patronaux, cotisations sociales, et autres éléments.

Prenant en considération la grande quantité des textes légaux qui sont publiés en continuité (lois, décrets, ordonnances, arrêtés, etc), et sachant que le pourcentage d'articles qui vont impacter la paie sur la totalité d'un document est presque toujours réduit, la lecture de textes représente définitivement un processus à optimiser.

C'est là où EBP a décidé de faire intervenir l'intelligence artificielle, en développant un outil qui permettra aux consultants de paie de trier de façon "autonome" les articles de chaque texte légal, ce qui permet de sélectionner au sein du corpus les textes qui ont la plus grande probabilité de définir des règles de paie et d'attirer l'attention du gestionnaire de paie sur les textes en rapport avec ces règles. Cet outil est conçu comme une API qui pourra être accédé à partir d'une interface dédiée, mais qui pourra également être consommée par des autres outils internes de l'entreprise.

La data

Nos données

Sachant que les règles de gestion de paie sont extraites, sans exception, à partir des différents textes légaux en cours de validité, les données à interpréter et étudier seront issues de ces textes. Une étude du format commun aux lois, décrets et autres, nous a permis de déterminer que les unités minimales et indivisibles à étudier sont les articles. Même si chacun des articles contient des informations qui peuvent être liées aux autres, chaque article correspond à une unité d'information qui doit être interprétée en entier.

Nos sources

Il est impératif que les textes à étudier soient totalement fiables, afin de garantir que le résultat du paramétrage de la paie soit correct et conforme aux dispositions en vigueur. Pour cela, nous allons prendre en considération uniquement des textes téléchargés à partir d'une source officielle : legifrance.gouv.fr. *Légifrance* est le site internet officiel du gouvernement français pour la diffusion des textes législatifs et réglementaires et des décisions de justice des cours suprêmes et d'appel de droit français. En plus, est un site d'accès libre, sous-titré « Le service public de la diffusion du droit », qui présente ou renvoie à la totalité des institutions et administrations concernées et à tous les textes encore en vigueur. L'utilisation exclusive des documents issus de cette source nous permettra de nous assurer qu'il n'y aura pas des fausses informations ou textes qui ne correspondent pas au besoin, car tous ces textes ont été déjà vérifiés par une équipe d'experts dans le domaine du droit. Finalement, cela fera sorte de garantie que le format du document est toujours l'attendu par notre application, sans des informations ajoutées, analyses subjectives, ou autres.

Dans un premier terme, l'idée était de présenter à l'utilisateur une interface qui lui permettra le téléchargement des documents en version PDF afin de les traiter à partir de ce format. Cependant, vu l'importance de pouvoir assurer le contenu du document, il n'est pas envisageable l'option de permettre à l'utilisateur le téléchargement du document pour son propre compte. L'idée, alors, était de servir de passerelle pour ces téléchargements, donc proposer les fichiers à l'utilisateur, les récupérer directement sur le site de Legifrance, et les stocker dans les dossiers propres à l'API.

Un étude du sujet nous a permis de constater que sur le site de Legifrance, aucun lien de téléchargement n'est disponible. Nous avons pensé, alors, à récupérer ce lien de téléchargement à partir du code HTML de chaque page correspondante à chaque document, mais le lien présent, quand utilisé sur une autre fenêtre ou depuis un autre site, ne redirectionne pas vers le document concerné mais télécharge un pdf vide à la place.



La solution proposée ensuite a été de demander à l'utilisateur de télécharger le document depuis son explorateur internet, puis uploader dans notre système le document PDF obtenu préalablement depuis le site, et finalement de notre côté vérifier la présence du metadata ou d'une signature électronique qui puisse valider la fiabilité de ce document et qu'il n'avait pas été modifié ou altéré. Malheureusement, aucune metadata ou signature n'a été trouvée dans les document téléchargés.

Finalement, nous avons trouvé la solution définitive pour combler notre besoin : une API REST mise à disposition par Legifrance via le portail PISTE qui, après inscription sur la plateforme et suite à l'obtention d'un token personnalisé, met à disposition des méthodes pour récupérer en format JSON tous les documents légaux qui puissent nous être utiles d'une façon directe et, surtout, fiable. Cette fiabilité des données, conséquence directe de la fiabilité de la source, nous permettra de les utiliser sans avoir besoin ni de vérifier l'existence des valeurs aberrantes ou de data incomplète, ni de faire une présélection des données à traiter : tous les textes téléchargés seront utilisés pour notre étude.

Malheureusement, il n'existe pas un type d'appel qui récupère l'ensemble des documents, donc il est nécessaire de télécharger un objet json contenant son structure (nom, date, liste des section et d'articles, entre autres informations) et, finalement, en utilisant une autre appel à l'API, télécharger le contenu de chacun des articles. Cette procédure, même si c'est un peu longue en termes de temps mais aussi de ressources, est définitivement la méthode la plus sécurisée pour l'obtention de cette data.

La portée de ce projet va se concentrer uniquement sur l'analyse et le traitement des Codes en vigueur. Des fonctionnalités pour inclure d'autres types de documents législatifs seront proposées et développées ultérieurement.

Les méthodes de l'API REST de Legifrance pour l'obtention des données sont :

POST	<code>/consult/code</code>	Contenu texte type CODE	
POST	<code>/consult/getArticleByCid</code>	Contenu des versions d'un article	

Le prétraitement et la sauvegarde

Une fois les textes obtenus à partir de l'API, nous allons les répertorier dans une table de notre base de données, en enregistrant le nom du fichier, son ID interne, sa date de publication ainsi qu'une valeur indiquant si le document a déjà été traité ou pas. Ensuite, une procédure sera automatiquement lancée afin de récupérer la liste des articles, télécharger le contenu de chacun. et l'enregistrer également dans une autre table de notre base des données. Pour chaque article nous allons sauvegarder le nom, ses 2 ID's internes à l'API de Legifrance, le texte brut et l'id du document de source.

L'Intelligence artificielle

La préparation des données

Après avoir récupéré le document et sauvegardé ses articles, l'utilisateur aura la possibilité de démarrer le traitement NLP des données. Dans le cadre d'un document législatif, et vu que l'objectif final de notre système d'IA est de les classer les articles légaux selon son domaine d'application et ne pas selon son sentiment général, il sera nécessaire de garder tous les mots, sans enlever même celles considérées vides, communément appelées "stop words", celles qui n'apparaissent que peu des fois, ou celles qui n'ont pas une longueur minimale prédéfinie. Tous les mots devraient être conservés intégralement. Nous n'aurons pas non plus à lemmatiser notre texte. Les seuls pré-traitements à faire seront la suppression des signes de ponctuation et des espaces vides multiples, ainsi que la séparation de mots composées et unies par un tiret.

Apprentissage supervisé vs. non supervisé

Pour développer un modèle d'IA basé sur l'apprentissage supervisé, il est nécessaire d'avoir une bonne quantité des données étiquetées qui puissent être utilisées pour entraîner le modèle et ensuite pour le tester, tout avant de pouvoir l'utiliser pour la prédiction des nouvelles valeurs ou étiquettes. Dans notre cas d'étude, nous n'avons pas accès à une telle source des données, c'est pourquoi la seule option c'est de créer un modèle basé sur l'apprentissage non supervisé.

La représentation du texte

Afin de pouvoir utiliser un modèle d'IA pour classer les articles, il est nécessaire de les représenter sous la forme d'une matrice numérique, qui puisse être compréhensible par l'ordinateur. Pour cela, ils existent plusieurs méthodes connues, notamment :

- Sac de mots (qui utilise l'algorithme TF-IDF)
- Représentation mot à mot
- Word2vec
- BERT

Une représentation du texte en sac de mots considère le texte comme un sac où tous les mots sont en vrac, sans prendre en compte sa position, fréquence ou les relations entre eux. En plus, cette représentation se limite aux mots les plus caractéristiques, laissant de côté tous les mots sans intérêt (articles, conjonctions, pronoms, adverbess, etc.). Comme nous l'avons déjà vu, le fait de ne pas traiter tous les mots peut sérieusement impacter la compréhension d'un texte légal, et c'est pour cette raison qu'utiliser un algorithme TF-IDF ne nous permettra pas d'avoir une représentation du texte adaptée au besoin.

La représentation mot à mot permet de numéroter chaque mot dans un dictionnaire, en ajoutant à des numéros issus d'autres caractéristiques (taille du mot, rang du mot dans la phrase, numéro de l'étiquetage grammatical). Ainsi, la place du mot dans la phrase est respectée. Contrairement au sac de mots, tous les mots y sont représentés, avec plusieurs caractéristiques pour chaque mot.

Le sac de mots et la représentation mot à mot prennent en compte chaque mot dans son individualité, sans considérer sa relation avec les autres. En revanche, ils existent d'algorithmes de "word embedding", qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels. Le word embedding repose sur une théorie linguistique connue sous le nom de "Distributional Semantics", qui considère qu'un mot est caractérisé par son contexte, c'est-à-dire, par les mots qui l'entourent. Ainsi, des mots qui partagent des contextes similaires partagent également des significations similaires. Word2vec et BERT font partie de ce type d'algorithmes.

Word2vec génère un seul vecteur par mot, indépendamment du contexte où le mot est utilisé, tant que BERT génère une représentation du texte qui permet chaque mot d'avoir plus d'un vecteur, basé sur le contexte d'utilisation. Cependant, en raison de sa complexité, BERT ne peut pas être utilisé pour la représentation de textes longs : cet algorithme est incapable de les traiter dû à son incrémentation quadratique de la consommation de temps et de mémoire. Normalement BERT est utilisé pour des textes de maximum 512 tokens (mots). Dans un article de loi, il y a normalement beaucoup plus de 512 mots. En prenant le Code du travail en vigueur à cette date, nous avons pu constater que l'article le plus long contient 17 205 mots.

Il y a des approches qui proposent la séparation d'un texte en plusieurs sub-textes d'un maximum de 512 tokens pour son analyse avec Bert. Cependant, la complexité de cette pratique pour des textes si longs comme nos articles ne justifie pas cette implémentation. Additionnellement, comme déjà expliqué, dans notre cadre légal il est inconcevable la division d'un article, car cela peut changer entièrement sa signification.

C'est pour cela que pour la représentation numérique de nos articles, nous allons utiliser Word2Vec comme algorithme.

Il existe des modèles Word2Vec pré-entraînés du domaine public, notamment ceux de Google. Le modèle Word2Vec de Google a été formé sur le jeu de données Google News (environ 100 milliards de mots), et est utilisé par plusieurs applications, telles que les moteurs de recommandation, la découverte des connaissances, et également appliqué dans les différents problèmes de classification de texte. Selon la manière dont le modèle apprend, Word2Vec est classé en deux approches :

- Continuous Bag-of-Words (CBOW)
- Skip-gram model

Le modèle CBOW (Continuous Bag-of-Words) apprend le mot cible en fonction des mots voisins, tandis que le modèle Skip-gram apprend les mots voisins en fonction du mot cible. Dans notre cas particulier, nous allons utiliser le modèle CBOW afin d'analyser le contexte (l'article) et trouver le mot cible (l'étiquette ou, dans notre cas, le champ d'application de l'article).

Le modèle

Une fois que les textes légaux seront proprement représentés dans un format numérique à l'aide du Word2Vec, la prochaine étape c'est de les diviser en deux partitions, ou clusters, ce qui nous permettra d'avoir deux groupes différents de textes, cela veut dire, deux champs d'application, ou deux "comportements" différenciés. Ce processus de séparation de la data en plusieurs catégories est, dans le cadre de l'Intelligence Artificielle, appelé "clusterisation".

Les deux algorithmes de clustering les plus populaires qui peuvent être utilisés sont K-means et DBSCAN. Ces algorithmes permettent de découvrir des structures cachées dans les données en identifiant des groupes avec des caractéristiques similaires.

K-means est un algorithme de clustering basé sur les partitions, qui recrée chaque donnée de l'ensemble dans un et un seul des nouveaux clusters formés. Une donnée ou un point de données est attribué à un groupe spécifique à l'aide d'une mesure de distance ou de similarité. Cet algorithme partitionne tous les points de la data en une quantité prédéfinie de groupes de similarité, et cette similarité entre points est généralement mesurée à l'aide de la distance euclidienne.

DBSCAN, de son côté, est un algorithme de clustering basé sur la densité. L'algorithme améliore les régions avec une densité suffisamment élevée en clusters et découvre des clusters de structure arbitraire dans des bases de données spatiales avec du bruit. Il définit un cluster comme un ensemble maximal de points connectés par densité, qui représente un ensemble d'objets connectés entre eux. Chaque objet non contenu dans un cluster est considéré comme du bruit.

Même si DBSCAN peut s'avérer plus précis et extrêmement efficace pour détecter les valeurs aberrantes et gérer le bruit, plusieurs études ont déterminé que K-means est beaucoup plus efficace avec de larges volumes de données. En addition, DBScan analyse la densité de la data et génère une quantité indéterminée de clusters, tant que K-means prend cette quantité comme paramètre d'entrée.

Pour des ensembles de données très importants, il existe une alternative au K-means, beaucoup moins gourmande en ressources : Mini Batch K-means. L'avantage de cet algorithme est de réduire le coût de calcul en n'utilisant pas tout le jeu de données à chaque itération mais un sous-échantillon de taille fixe. Cette stratégie réduit le nombre de calculs de distance par itération au prix d'une qualité de cluster inférieure. Cependant, dans des environnements avec une quantité réduite de clusters, Mini Batch K-means a prouvé être aussi efficace que son "grand frère": K-means. Vu que notre intérêt est de diviser toutes nos données en uniquement deux parties, cet algorithme représente l'option idéale pour la suite de notre applicatif.

En addition à la quantité de clusters à créer, Mini Batch K-means peut prendre un paramètre appelé Batch Size, qui représente le nombre d'images utilisées pour entraîner le réseau. Plusieurs études ont montré qu'en utilisant Mini Batch K-means, les batch sizes plus grandes prennent moins de temps à entraîner, mais sont beaucoup moins précises. L'erreur arrive à son point le plus petit autour des valeurs proches à 32 pour la taille du batch, mais le temps de calcul augmente considérablement. La documentation officielle de Mini Batch K-means propose, pour des calculs plus rapides, de définir la taille du batch à $256 * \text{nombre de cœurs}$, afin d'activer le parallélisme sur tous les cœurs. Nous avons choisi une valeur de 512, ce qui nous permettra d'utiliser 2 cœurs en parallèle, sans trop compromettre les résultats. Pour les futures versions de cette application, une fois que le sujet d'étude sera élargi afin d'inclure d'autres types de documents législatifs (lois, dérogations, etc), un étude sera proposé afin de de déterminer la taille du batch la plus adaptée.

Les technologies

Comme déjà indiqué, notre applicatif sera créé sous la forme d'une API, qui sera principalement consommée pour des autres outils internes. Cette API a été conçue sur la base de Fast API, ce qui permettra aux utilisateurs d'avoir toujours l'accès en direct au Swagger, et de pouvoir tester les différentes routes proposées. En plus, une interface développée en Angular sera proposée, afin de permettre aux collaborateurs de l'utiliser, en attendant que les autres outils soient adaptés. Toutes les données seront stockées dans une base MySQL qui comprendra uniquement 2 tables : une pour les articles et une autre pour les utilisateurs. Il n'est pas nécessaire d'inclure une gestion d'utilisateurs, vu que toutes les politiques de sécurité de l'entreprise sont gérées à partir d'un Active Directory, ce qui gère les droits d'accès aux sites et aux BD par groupes d'utilisateurs.

Pour l'interface, une seule maquette a été réalisée, qui correspond au template de base. Tous les contenus seront affichés dans des tableaux à l'intérieur de cette page.

Notre page d'accueil affichera la liste des Codes légaux disponibles pour son téléchargement depuis le site de Legifrance, ce qui permettra aux utilisateurs d'avoir toujours un visuel sur les nouveaux documents disponibles, faisant sorte d'un système de veille technologique. A côté de

chaque document il sera affiché la date de publication, ce qui permettra de comparer avec la version enregistrée en base des données, afin de déterminer s'il est nécessaire de le télécharger à nouveau. Le téléchargement du document et son stockage dans la base de données, seront possibles directement à partir de cet écran.



L'interface permettra également à l'utilisateur d'avoir un visuel sur les documents déjà processées à l'aide des algorithmes du NLP, et de démarrer le traitement pour ceux qui n'ont pas encore été analysés. Il sera également possible d'afficher les clusters assignés à chaque article d'un document.

Une future version de cette application pourrait également inclure un module pour télécharger la nouvelle version d'un document et la comparer avec cela en base des données, afin d'informer à l'utilisateur quels articles ont changé par rapport à la dernière version, ce qui permettra de faciliter encore plus son traitement et d'optimiser le temps. Il est également prévu d'inclure des interfaces qui permettront la consultation des articles par cluster, et pas uniquement par document. Cependant, ce sont des fonctionnalités qui correspondent, dans notre planning interne, aux stades beaucoup plus avancés du projet.