Classification de texte pour le diagnostic d'intervention

Adueni Adjoba Eunice AKANI
Aix-Marseille Université
akanieunice@hotmail.fr

Romain GEMIGNANI
Chef de Projets Smart Grid & Mobilité Elec
romain.gemignani@enedis.fr

Abstract

Les méthodes de machine learning sont devenues de plus en plus utilisées pour la résolution de problèmes de classification ou de régression. Basées sur des approches statistiques, elles ont pour rôle de résoudre implicitement les tâches en s'améliorant grâce à leur apprentissage sur les données.
De son côté, Enedis, société de distribution d'électricité de référence, se doit d'améliorer en permanence ses processus et ses performances, tout en restant une entreprise à la pointe de la technologie. C'est dans cette politique qu'elle intègre de plus en plus l'intelligence artificielle dans son travail quotidien.

Dans le cadre de notre stage, nous nous sommes focalisés sur les Bons de Travaux générés par les agents d'Enedis avant une intervention. Le but de notre stage était d'appliquer des méthodes de TAL[1] à la pointe de l'état de l'art pour l'analyse et l'extraction d'informations des interventions réalisées sur le réseau public de distribution, en déterminant les principales causes d'interventions vaines, puis d'estimer la durée des interventions. En plus des Bons de Travaux, nous parlerons entre autre de l'application industrielle de certaines méthodes de TAL pour une autre utilisation, telle que le routage de réclamations des clients.

1 Introduction

Le traitement automatique de la langue naturelle est une branche de l'intelligence artificielle qui utilise des méthodes de machine learning pour trouver des solutions à une variété de tâches complexes telles que la création de traducteur automatique, la génération de texte, l'analyse des sentiments, etc.

Enedis, grande entreprise de distribution d'électricité, possède un énorme flux de données textuelles exploitables par des algorithmes de machine learning. C'est dans ce contexte que notre stage de fin d'étude se situe. Au cours de la journée des agents Enedis effectuent plusieurs interventions qui peuvent être "réalisées" ou "vaines". Afin de minimiser le nombre d'interventions vaines, l'entreprise souhaiterait connaître les causes de celles-ci et aussi optimiser la durée de chaque intervention. Chaque intervention effectuée donne naissance à une fiche appelée "Bon de Travaux" (BdT) qui permet de la suivre à chaque étape. Ainsi, notre travail sera d'analyser les différents BdT d'Enedis afin d'en ressortir des résultats permettant une meilleure compréhension donc de rechercher des pistes d'amélioration, voire d'optimisation.

Pour ce faire, dans un premier temps nous présenterons les différents types de BdT à travers une analyse de données. Ensuite, nous appliquerons quelques algorithmes de TAL sur les bons de travaux et enfin, nous parlerons des tâches annexes effectuées pendant le stage.

2 Les Bons de Travaux (BdT)

Pour chaque intervention, un BdT est créé et mis à jour selon plusieurs états. Les différents états sont : Initialisé - Préparé - Signé - Reçu - En cours - Réalisé - Soldé - Clôturé . Un BdT peut être annulé pendant l'un des états ou en point d'arrêt en cas de danger.

[1] TAL: Traitement Automatique de la langue

Attributs	Description
Nature de la BdT	Correspond au domaine qui doit réaliser l'intervention
Rubrique	C'est un champ qui permet de dire dans quel domaine l'intervention se fait
Catégorie du BdT	C'est un champ comme la rubrique. Il indique si le BdT est programmé ou non (en temps réel, prévisionnel)
Objet	Il est indicatif comme l'objet d'un mail. L'agent qui le rempli peut utiliser un modèle existant ou en créer un autre
Description des travaux	C'est la description des travaux à effectuer. Elle est renseignée par celui qui construit le BdT
Nombre d'agents	Le nombre d'agents intervenant sur le BdT
Durée estimée	C'est la durée estimée pour la réalisation de l'intervention; Elle est en lien avec l'objet de la BdT
Méthode de travail	Permet de signifier si l'intervention doit se faire sous tension ou hors tension
Type réseau	Indique s'il s'agit d'une intervention sur un poste HTA (haute tension) ou BT (basse tension)
Code postal	Identifie le lieu où se fera l'intervention
Type de BdT	Décrit l'objet sur lequel les agents doivent travailler
Taux de réalisation	C'est le champ qui indique si l'intervention a été réalisée et donne également le pourcentage de réalisation

Table 1: Description des attributs.

2.1 Description des données

Il existe deux types de BdT : les BdT comptage et les BdT réseau. Pour les différentes expériences sur les BdT réseau, nous avons travaillé sur des BdT extraits sur 3 années successives (2017, 2018 et l'année en cours 2019). Les différents champs du formulaire n'étant pas des champs obligatoires nous étions confrontés à des données contenant de nombreux champs vides.
Les BdT contenaient plusieurs attributs mais nous avons utilisé que ceux de base c'est à dire ceux à l'initialisation des BdT. La description des différents attributs est consignée dans la Table 1.

2.2 Description des tâches effectuées

Les différentes tâches sont :

- Prédire la probabilité qu'une intervention soit vaine : Dire si une intervention à une forte probabilité d'être vaine peut réduire le nombre de déplacements inutiles.
- Déterminer la cause des interventions vaines.
- Inférer la durée d'une intervention : Le but étant d'indiquer si la durée estimée est conforme à l'intervention qui doit être faite.

Etant donné la nature des données, il est important de savoir quels sont les features[2] importants pour la prédiction ainsi que la faisabilité de la tâche. Pour ce faire, nous avons utilisé ICSIBoost (Favre et al., 2007) et Bonzaiboost (Laurent et al., 2014), des classifieurs de texte qui sont principalement basés sur des arbres de décision.
ICSIBoost combine un ensemble de classifieurs faibles pour former un classifieur fort. Il est basé sur la technique de boosting d'Adaboost avec un niveau de décision. Bonzaiboost quant à lui fait un meta apprentissage des arbres de décision; il implémente les algorithmes de Boosting (AdaBoost.MH) et bagging (RandomForest). Les deux classifieurs peuvent combiner des attributs de type texte, discret,

[2]Feature : propriété mesurable individuelle ou caractéristique d'un phénomène observé

État	Clôturé	Annulé	Soldé	Signé	Préparé	Reçu	En cours	Réalisé	Initialisé
Total	679 803	273 475	27 094	10 256	5352	2406	967	349	10

Table 2: Nombre de BdT par état.

Données	100%	vaine	non effectuée	75%	50%	25%	nan
Total	668 654	7 185	6 526	4 822	4 367	2 523	12 820

Table 3: Classes à prédire : Taux de réalisation

nombre. Ils prennent également des données avec des attributs manquants.

2.3 Analyse des données

Nous avions au total 999 712 BdT sur les 3 dernières années. Le tableau 2 illustre le nombre de BdT par état. Nous n'avons utilisé que les BdT clôturés et soldés car pour notre tâche qui était de prédire le champ "taux de réalisation", c'est seulement à ces deux états que ce champ était rempli. Ainsi, il ne restait que 706 897 BdT. Le champ à prédire "taux de réalisation" contenait les valeurs 100%, 75%, 50%, 25%, vaine, non effectuée et nan[3] (non renseigné). Le tableau 3 contient le nombre de BdT ayant ces différentes valeurs. Nous remarquons que la classe vaine est la classe minoritaire.

Étant donné que le taux de réalisation est le champ à prédire, il était convenable de supprimer les valeurs "nan" . En les supprimant, nous nous sommes donc retrouvés avec 694 077 BdT. Nos features de départ peuvent contenir des valeurs nulles; Le tableau 4 présente le pourcentage de valeur nan dans chaque feature et nous montre que environ 20% des BdT ont le champ Description_Travaux sans valeur.

Feature		% de nan
Nature_BdT	100	0
Rubrique	100	0
Categorie_BdT	80.05	10.95
Objet	100	0
Description_Travaux	80.48	19.52
Nombre_Agent	100	0
Duree_estimee	100	0
Methode_travail	95.4	4.6
Type_reseau	95.14	4.86
Commune_code_postal	98.08	1.92
Taux de realisation	100	0
Type Bdt	100	0

Table 4: Pourcentage de champ vide ou rempli pour chaque feature.

2.4 Expérience et résultats

2.4.1 Prédiction du Taux de réalisation

Pour effectuer cette tâche, nous nous sommes concentrés sur les données dont les valeurs du taux de réalisation sont 100% et vaine. La classe vaine (minoritaire) était celle qui nous intéressait le plus pour nos analyses. Ainsi, nous avons utilisé ICSIBoost et Bonzaiboost pour faire ressortir les features importants. ICSIBoost ne pouvant entraîner un maximum de données, nous avons réduit les données de la classe majoritaire pour faciliter l'apprentissage des composantes de la classe minoritaire. Les résultats

[3]Nan : Not a number

obtenus avec ISCIBoost n'étant pas concluant, nous avons repris les mêmes tests avec Bonzaiboost. Les matrices de confusion (table 5) nous montre que le classifieur confond très souvent la classe "vaine" à la classe "100%".

ref/hyp	0	1
0	4609	582
1	519	534961

ref/hyp	0	1
0	110	1884
1	1023	132151

Table 5: Matrice de confusion sur les données "train" et "test". (A gauche, la matrice de confusion sur le train et à droite celle sur le test)

En développant les arbres de décision, Bonzaiboost offre la possibilité d'avoir le feature utile à la prédiction de la classe "vaine" ou "réalisé". Les features les plus sélectionnés pour la prédiction du taux de réalisation sont la description de l'intervention, l'objet et le code postal; ils ont été sélectionnés à 89.6% (voir Tableau 6).

Feature	pourcentage
categorie bdt	0.0%
code_postal	17.7%
nombre agent	2.2%
objet	50.3%
description	21.6%
rubrique	2.0%
type bdt	0.5%
type réseau	0.9%
duree estimée	3.8%
méthode travail	1.0%
nature	0.0%

Table 6: Pourcentage de sélection des features pour la prédiction du taux de réalisation

2.4.2 Détermination des causes vaines

Nous avons remarqué que la tâche (prédire le champ "taux de réalisation") est quasiment impossible avec les anciennes techniques. Il était donc nécessaire de connaître les causes des interventions dans le but de savoir si elles sont réellement prévisibles. Pour ce faire, nous avons effectué un clustering sur la description du compte rendu de chaque intervention.
Le clustering, méthode non supervisée (c'est à dire que les données ne sont pas labellisées) dans le but de regrouper des données en paquets, a été utilisé dans notre cas pour regrouper toutes les interventions vaines afin d'en extraire les principales causes.
L'algorithme de clustering utilisé était le "k-means" (k-moyenne). L'inconvénient de cet algorithme c'est qu'il faut connaître en avance le nombre de clusters. Il est donc impossible de connaître le nombre optimal de clusters. Cependant, il existe des méthodes utilisées de manière itérative qui peuvent indiquer si le nombre de clusters est optimal. Pour notre part, nous avons testé trois méthodes :

- La méthode Elbow consiste à chercher le point qui représente le coude sur la courbe du nombre de clusters en fonction de la variance. Le point représentant le coude est le nombre optimal de cluster. La variance se calcule comme suit :

$$V = \sum_{j} \sum_{x_i \longrightarrow c_j} D(c_j, x_i)^2 \tag{1}$$

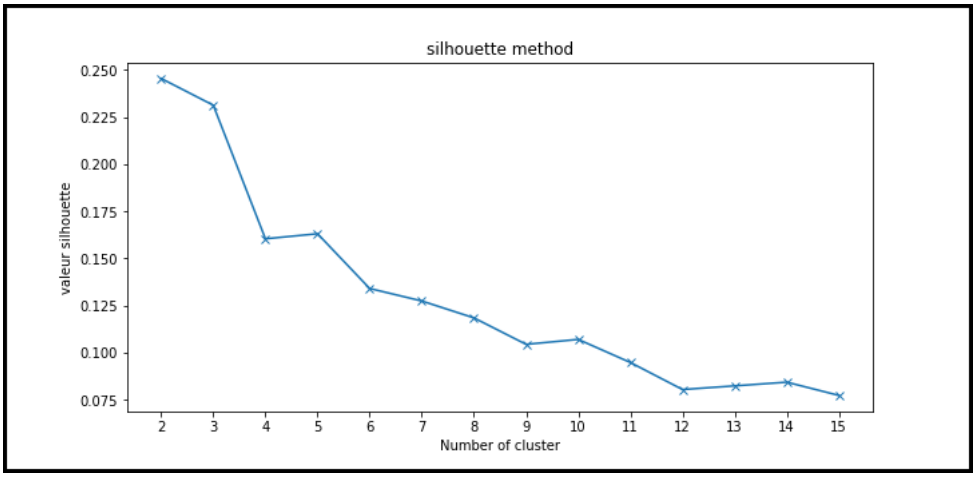

Figure 1: Courbe représentant le nombre de cluster en fonction de la taille de la silhouette

En utilisant, la méthode d'Elbow (voir Annexe B) sur nos données, il est difficile de déterminer la nombre optimal de clusters.

- La méthode silhouette permet de vérifier si chaque document est bien classé. La valeur de la silhouette pour une observation x_i est

$$S(i) = \frac{b_i - a_i}{\max(b_i, a_i)} \tag{2}$$

avec : $a_i = \frac{1}{n(C(i))} \sum_{x \in C(i)} d(x_i, x)$ et $b_i = \min_{C \in \mathscr{C} \setminus C(i)} \frac{1}{n(C)} \sum_{x \in C} d(x_i, x)$
La largeur de la silhouette est la moyenne de toutes les valeurs de la silhouette, elle est comprise entre -1 et +1. Elle doit être maximisée. L'utilisation de la méthode de la silhouette conduit à deux cluster car pour deux cluster, la valeur de la silhouette est maximale (Figure 1).

- Le critère de Davies-Bouldin est minimum lorsque le nombre de classes (de cluster) est optimal. Il permet de comparer les distances intra-cluster (homogénéité[4]) et les distances inter-cluster (séparation[5]).

$$DB = \frac{1}{C} \sum_{i=1}^{C} \max_{i \neq j} \frac{\sigma_i + \sigma_j}{d(C_i, C_j)} \tag{3}$$

Avec Davies-Bouldin, on obtient les mêmes résultats qu'avec la méthode de la silhouette c'est à dire 2 clusters (Figure 2).

Pour réaliser le clustering, on peut utiliser les librairies python nltk ou scikit-learn. Il est plus convenable, dans notre cas, d'utiliser la librairie nltk. Nltk est une bibliothèque python permettant un traitement automatique des langues. elle implémente également l'algorithme du k-means.

Représentation des données pour le clustering :
Le feature utilisé est la description du compte rendu. C'est un champ texte libre (rempli à la main). Il est donc parfois rempli avec de nombreuses erreurs et fautes d'orthographe. Nous avons utilisé une représentation "Word Embedding" pour le représenter. La représentation "Word Embedding" permet de faire correspondre un vecteur (de dimension à définir) de nombres réels à un mot. Elle représente les mots similaires par deux vecteurs relativement proches. Il est possible d'utiliser des embeddings pré-entraînés sur grand un corpus, cependant nous avons créé des vecteurs d'embedding en prenant en compte tous les textes dans le champ description puisque le langage utilisé est propre au domaine.

[4]homogénéité d'un cluster: la moyenne des distances de chacun des points contenus dans ce cluster.
[5]Séparation de deux clusters : c'est la distance entre leurs centroïdes.

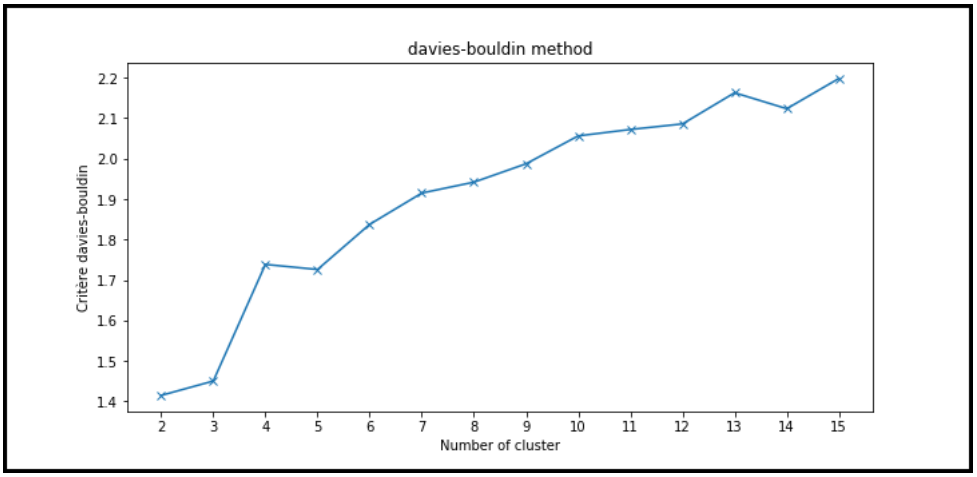

Figure 2: Courbe représentant le nombre de clusters en fonction du critère de Davies-Bouldin.

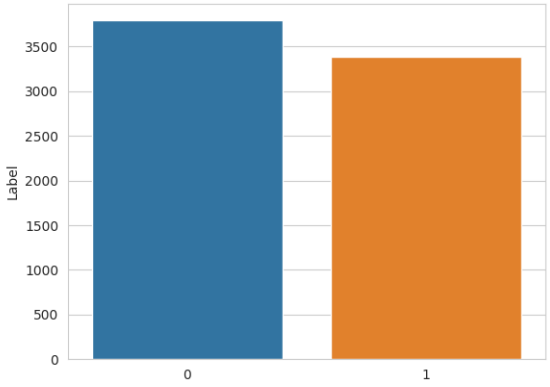

Figure 3: Nombre de données par cluster

Ainsi, grâce à la librairie de Facebook, fastText (Joulin et al., 2016) (Bojanowski et al., 2016), nous avons représenté chaque mot en un vecteur de 200 dimensions. Il est avantageux d'utiliser fastText car il génère également un embedding pour les mots hors vocabulaire sans oublié qu'il n'est pas sensible aux fautes d'orthographe. Pour retrouver le vecteur de chaque description (chaque phrase), nous avons moyenné les différents vecteurs d'embedding de chaque mot de la phrase.

Causes des interventions vaines :
Basés sur les résultats obtenus (Figure 1 et 2), nous avons fait l'algorithme des k-means avec 2 clusters.

$$\cos\theta = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{4}$$

La distance utilisée pour l'algorithme est la similarité cosinus, la même distance que celle utilisée pour la génération de vecteurs d'embedding. Elle est calculée comme l'indique la formule 4. Le nombre de données appartenant aux différents clusters est presque le même. Dans le cluster 0 nous avons environ 3600 données et environ 3500 données dans le cluster 1 (Figure 3). Dans le cluster 0, le mot ayant la plus forte occurrence est "reprogrammer" ; par contre dans le cluster 1 c'est "client" (Figure 4). Ainsi on peut déduire que les principales causes d'interventions vaines sont traduites par l'absence du client et le manque de temps qui implique une demande d'un nouveau créneau pour effectuer l'intervention.

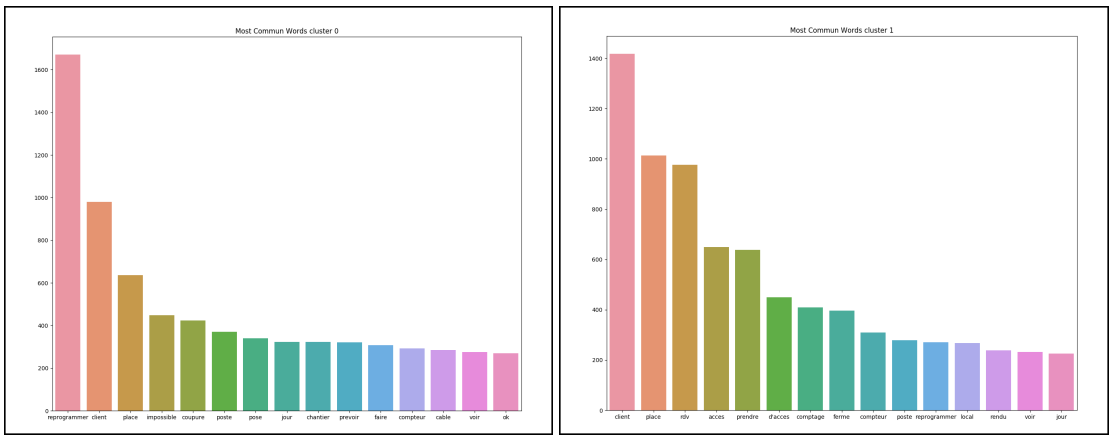

Figure 4: Les 15 mots avec une forte occurrence dans chaque cluster

Feature	pourcentage
categorie bdt	0.1%
code_postal	9.7%
nombre agent	0.6%
objet	30.9%
description	48.4%
rubrique	3.0%
type bdt	1.9%
type réseau	0.6%
duree estimée	3.9%
méthode travail	0.7%
nature	0.2%

Table 7: Pourcentage de sélection des features pour la prédiction de la durée

2.4.3 Inférence sur la durée d'une intervention

La durée d'une intervention dépend du champ "objet"; Elle peut être égale ou non à celle estimée. Il était donc important de vérifier si la durée estimée d'une intervention est réellement conforme. Nous avons donc créé 3 classes à partir de la durée estimée et la durée réelle d'une intervention : ecart = durée_travaux - durée_estimée . Les 3 classes sont :

- ecart > 0 implique Durée travaux $>$ Durée estimée : Dans ce cas, c'est un problème qui peut fausser la planification.
- ecart = 0 implique Durée travaux = Durée estimée : Dans ce cas, nous sommes conforme au planning
- ecart < 0 implique Durée travaux $<$ Durée estimée : Ne touche pas vraiment le planning mais peut être utile pour affecter un agent à d'autres interventions.

Dans un premier temps, nous avons utilisé bonzaiboost pour avoir les features importants à la prédiction de ces trois classes. Les données ont été divisées en deux datasets : 80% des données dans le dataset du train et 20% dans le dataset du test. Comme le montre le tableau 7 la description et l'objet sont les features les plus utilisés dans la prédiction; ils ont été sélectionnés à 79.3% pour la prédiction de la durée.

Prédiction de la durée : Résultat avec bonzaiboost

En inférence sur la durée (suffisante, peu ou trop), nous avons obtenu avec Bonzaiboost un F-measure de 99.77% sur les données train et 90.10% sur les données test. La F-measure sur la classe minoritaire

(durée travaux ¿ durée estimée) est de 98.86% sur le train et 82.23% sur le test. La F-measure sur les données test de la classe 1 (durée travaux ¡ durée estimée) est moins bonne que les autres (62.01%). Les résultats sur les données train et test sont consignés dans les tableaux 8 et 9 La F-measure est une

Label	HYP	REF	Correct	Error	Precision	Recall	F-Measure	Error Rate
0	27963	27776	27552	411	98.53	99.19	98.86	1.48
1	52191	52187	21919	272	99.48	99.49	99.48	0.52
2	376255	376446	375898	548	99.91	99.85	99.88	0.15
All	456409	456409	455369	1040	99.77	99.77	99.77	0.23
Macro F1								99.41
Exact Match								99.77
Multi-Label-Accuracy								99.77
Hamming Loss								0.08

Table 8: Résultat de l'exécution de bonzaiboost pour la prédiction de la durée sur les données train

Label	HYP	REF	Correct	Error	Precision	Recall	F-Measure	Error Rate
0	6861	7046	5718	1328	23.34	81.15	82.23	18.85
1	12527	12950	7899	5051	63.06	61.00	62.01	39.00
2	94715	94107	89187	5528	94.16	94.77	94.47	5.87
All	114103	114103	102804	11299	90.10	90.10	90.10	9.90
Macro F1								79.57
Exact Match								90.10
Multi-Label-Accuracy								90.10
Hamming Loss								3.30

Table 9: Résultat de l'exécution de bonzaiboost pour la prédiction de la durée sur les données test

mesure de la précision d'un test. Elle prend en compte à la fois la précision et le rappel du test. Elle se calcule comme suit :

$$F_1 = 2.\frac{precision.rappel}{precision + rappel} \tag{5}$$

Etant donné que les F-measure pour chaque classe avec les méthodes classiques étaient élevées, la fonction restante était de trouver un modèle neuronal adapter à la prédiction de la durée.

Utilisation de réseaux de neurones

Les résultats obtenus avec Bonzaiboost étaient concluant; Ainsi, nous avons testé des modèles neuronaux pour avoir de meilleurs résultats. En entrée de notre réseau, nous avons utilisé quatre features (l'objet, la description, le code postal et la durée estimée). les Les entrées du reseau neuronal ont été représentées de la façon suivante :

- Description des travaux : nous avons utilisé le même embedding que dans la partie 2.4.2.
- Objet : étant un champ texte, nous avons utilisé un word embedding (fastText) pour le représenter. La dimension de l'embedding était 100.
- Code postal et durée estimée : ce sont des champs de type entiers. Pour nos expériences, nous les avons standardisé afin d'améliorer la qualité des données, avant de les passer en entrée.

Notre sortie était un vecteur one-hot de 3 dimensions. La figure schématise notre modèle. Le modèle implémenté était constitué de couches Dense et de Convolution à une dimension.

A partir de celui-ci, nous avons obtenu 92% d'accuracy sur le dataset du test. La F-measure sur chaque classe était : 80% pour la classe 0, 55% pour la classe 1 et 95% pour la classe 2.

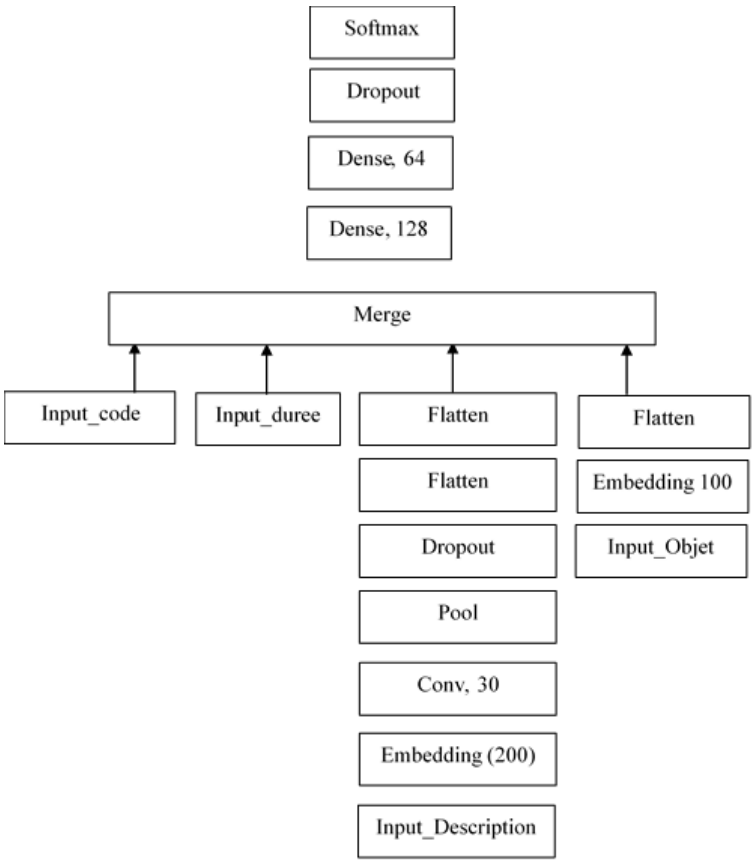

Figure 5: Modèle neuronal pour la prédiction de la durée

3 Les Bons de Travaux (BdT) comptage

Les BdT comptage sont les interventions faites en prenant en compte accessibilité d'un compteur. L'objectif est de prédire qu'une intervention sur un compteur sera vaine en prenant en compte l'historique sur ce compteur (son accessibilité, son fonctionnement, etc.).

3.1 Description des données

Les données ont à peu près la même structure que les données des BdT vues précédemment. La première différence entre les deux est le fait que celles des comptages sont remplies avec beaucoup plus de rigueur (champ obligatoire dans le formulaire) donc il y a moins de champs vide. Ensuite, il y a de nombreux champs en plus sur les BdT comptage qui permettent de montrer si une intervention a déjà été faite. En plus, sur les BdT comptage il y a plus de détails sur les causes d'une intervention vaine (existence d'un champ aléa). La description des champs est dans le tableau 10

3.2 Description des tâches effectuées

Les différentes tâches étaient :

- Prédire la probabilité de faisabilité d'une intervention en prenant en compte l'historique sur les compteurs, leur état.
- Inférer l'accessibilité du compteur lorsque ce champ n'est pas rempli.
- Créer un portail pour l'utilisation du modèle.

Nous avions au total environ 17 000 BdT comptage. Parmi ces 17 000, environ 15 500 ont été réalisés à 100% et environ 1 500 étaient des interventions vaines (Figure 6).

Feature	Description
Objet	C'est la desription de l'intervention à réaliser. C'est un champ texte libre.
Type Bdt	Correspond au type de la Bdt c'est le même champ que dans les BdT réseau
Accessibilité intervention	Deux valeurs possibles oui ou non; Il permet d'indiquer si un compteur est accessible
Année de fabrication	C'est l'année de fabrication du compteur
Type de compteur	Indique le type de compteur sur lequel l'intervention est faite
Base de comptage (BC)	Indique le lieu où est basé le compteur.
Communication	permet de savoir comment le compteur communique, le type de communication
Résultat de l'intervention	correspond au champ taux de réalisation dans les BdT réseau
Code INSEE	correspond au code postal
Durée estimée	C'est la durée prévisionnelle de l'intervention.

Table 10: Description des attributs des Bdts Comptage

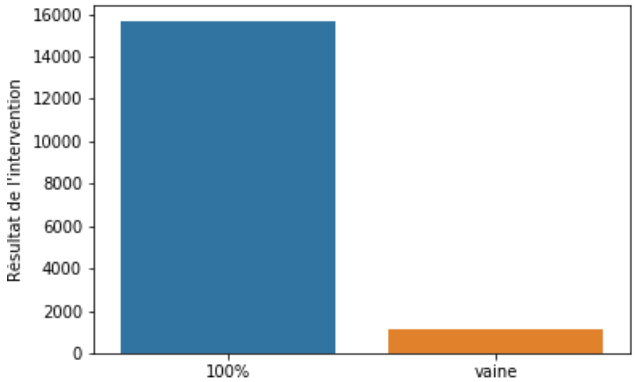

Figure 6: Histogramme présentant le nombre de BdT comptage pour chaque classe.

Label	HYP	REF	Correct	Error	Precision	Recall	F-Measure	Error Rate
non	865	885	420	465	48.55	47.46	48.00	52.54
oui	1742	1722	1277	465	73.31	47.16	73.73	27.00
All	2607	2607	1697	910	65.09	65.09	65.09	34.91
Macro F1								60.86
Exact Match								65.09
Multi-Label-Accuracy								65.09
Hamming Loss								17.45

Table 11: Résultat de l'exécution de bonzaiboost pour la prédiction de l'accessibilité du compteur

Il n'y a que 2 features qui sont des champs textes remplis à la main. Les autres sont des champs à valeurs discrètes connues au départ.

3.3 Expérience et résultats

3.3.1 Prédiction de l'accessibilité du compteur

En utilisant bonzaiboost pour la prédiction, on obtient 97% sur le dataset du train (cf. Annexe C) et environ 65% pour la F-measure sur le dataset du test. Ce qui veut dire qu'avec un réseau neuronal, on peut améliorer la prédiction. Bonzaiboost arrive à prédire la classe 'oui' avec 73% de précision par contre il est moins bon sur la classe 'non' sur laquelle il est à 47% de précision (voir table 11). Nous travaillons présentement sur un modèle neuronal pour la prédiction de l'accessibilité d'un compteur.

3.3.2 Prédiction de la probabilité d'intervenir

Il s'agit de la même tâche que celle faite au point 2.4.1. Nous avons déterminé les features importants pour la prédiction du champ Résultat de l'intervention (cf Annexe C). Le code postal (code INSEE), l'objet et le type de BdT ont été choisis à plus de 75% pour la prédiction du résultat de l'intervention. Nous avons effectué une classification binaire à l'aide de bonzaiboost mais les résultats obtenus ne correspondaient pas à nos attentes (cf. Annexe C). En test, nous avons une F-measure de 96.40% sur la classe 1 (réalisée à 100%) mais que 24.22% sur la classe 0 (vaine) avec une précision de 50% sur cette classe. Notre classifieur prédit presque tout le temps la classe majoritaire (la classe 100%). Ces résultats s'expliquent par le fait que la classe vaine est minoritaire (nous n'avons assez d'exemples de cette classe).

4 Autres tâches : Routage de réclamations

4.1 Classification des réclamations

Lorsqu'une intervention est faite, le client peut en être satisfait ou non. Lorsque celui-ci ne l'est pas, il adresse certaines fois une réclamation à Enedis. Ces réclamations sont alors déposées dans une grande base de données puis extraites régulièrement par des agents d'Enedis qui les lisent puis les routent aux services compétents pour les traiter. Ce procédé de lecture et de routage demande assez de temps et de ressources humaines.

Pour réduire ce temps de traitement par les agents d'Enedis, nous avons créé un outil qui permet de rediriger les différentes réclamations aux différents services grâce à des méthodes d'intelligence artificielle plus précisément de traitement automatique de la langue.

Dans un premier temps, il était question de déterminer une représentation pour les données en entrée qui sont les messages contenus dans les réclamations et des données en sortie qui sont les services adéquats. Pour ce faire, nous avons utilisé fastText que nous avons entraîné sur plus de 100 000 réclamations provenant de tout le territoire national afin obtenir des vecteurs d'embedding de dimension 200. Les sorties sont représentées en vecteur one hot (qui permet de représenter chaque classe en binaire de telle sorte que la représentation n'ait qu'un seul chiffre à 1). Ensuite, il fallait trouver un modèle efficace et léger pour la prédiction du service qui doit être embarqué sur un serveur de 2 Go de RAM, 2.29 Ghz

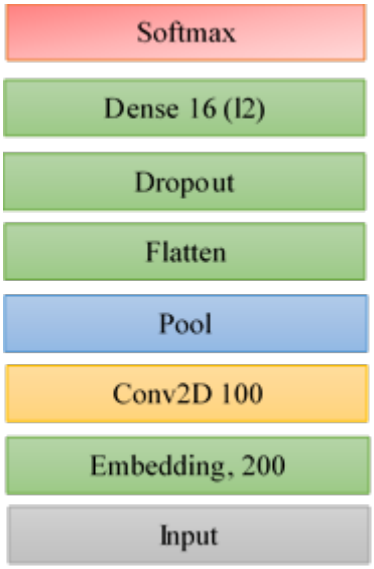

Figure 7: Modèle neuronal pour le routage des réclamations dans le bon service

(Intel Xeon) possédant un seul cœur. Le modèle devait donc répondre aux contraintes matérielles.

Nous avons utilisé un réseau neuronal convolutif (CNN) d'après (Kim, 2014) . Il s'agissait de retrouver des patterns dans le texte des réclamations afin de les router. Les CNN limitent le nombre de connexions entre un neurone et les neurones de la couche adjacente. Leur but premier est de prétraiter de petites quantités de données en entrée dans le but de déterminer des similitudes. Le modèle prenait en entrée un input qui est d'abord tokenisé puis on y ajoute un padding grâce à des fonctions de la librairie python keras. Cet input est de dimensions 370 et est injecté dans une couche d'embedding keras dont les poids sont initialisés grâce à l'embedding fait sur le texte des réclamations à partir de fastText. La couche embedding est suivie de la couche convolutionnelle à deux dimensions (taille du filtre par la dimension du vecteur d'embedding) suivie d'un Dropout et d'une couche de régularisation l2 pour éviter l'overfitting (le surapprentissage) et enfin suivie de la couche d'activation pour la prédiction des 5 classes.
La figure 7 illustre le modèle neuronal utilisé.
Les paramètres de notre modèle sont :

- Dimension de l'embedding : 200
- Dropout rate : 0.5
- Nombre de filtre du CNN : 100
- Taille des filtres CNN : 3
- Batch size : 32
- Fonction d'activation : softmax
- Fonction loss : categorical cross entropy
- Optimiseur : Adam
- Métrique : Accurancy

L'évaluation sur le dataset du test nous donne une accuracy de 90%. Le but à long terme est d'améliorer ce pourcentage.

4.2 Détection d'erreurs

Lorsque le champ type de réclamation est mal rempli, le traitement de cette réclamation devient plus long. Cette erreur affecte surtout les clients "marchés d'affaires" pour lesquelles les délais de réponse se doivent d'être rapides. Ainsi, nous avons implémenté un modèle neuronal qui détecte les erreurs au niveau du type de la réclamation s'il en existe. Ce modèle prend en entrée la description de la réclamation représentée avec le même embedding que dans le point 4.1. Le modèle de prédiction utilisé est le même. Il détecte avec une accuracy d'environ 80% les erreurs.

4.3 Portail des réclamations

Pour l'utilisation de nos différents modèles, nous avons conçu un portail web sur lequel, les agents peuvent voir les différentes réclamations et leurs routages. Le scénario pour le routage de la réclamation est le suivant:

- L'utilisateur dépose un fichier de type excel ou csv.
- Notre réseau neuronal effectue la prédiction de la route avec une certaine probabilité de confiance. Deux cas se présentent :
  - Si la probabilité est inférieur à un certain seuil de confiance, il est demandé à l'utilisateur de donner la destination manuellement.
  - Sinon l'outil l'envoie au bon service.
- Il effectue un apprentissage périodique (chaque fin de semaine) afin d'apprendre les caractéristiques des nouvelles réclamations.

5 Conclusions et perspectives

Les différentes tâches effectuées étaient basées sur la qualité et la quantité des données. Pour les BdT réseau, nous avions une quantité importante de données mais elles n'étaient pas remplies avec rigueur. Par contre, pour les BdT comptage, il y avait peu de données mais celles-ci étaient remplies avec rigueur. La difficulté de la tâche "prédire la probabilité d'effectuer une intervention" s'explique par le fait que les données ne soient pas équitablement distribuées dans les deux classes (vaine et réalisée). Les résultats obtenus sur la prédiction de la durée peuvent être d'un grand avantage dans la planification des interventions.

Notre étude a été réalisée dans le cadre du stage de fin d'étude du Master 2 et avait pour objectif principal d'extraire des données, de les analyser afin de trouver une application de TAL à ces données.

Ce stage fut pour nous un moyen d'appliquer des méthodes d'intelligence artificielle sur des données réelles et de comprendre l'importance de l'analyse et la préparation de données avant application d'algorithme. Toutes les tâches ne sont pas toujours faciles à implémenter à l'aide de réseaux neuronaux. Il est donc important d'évaluer leurs difficultés. Aussi, trouver un modèle adapté à la tâche est une des difficultés majeur du deep learning.

Notons que les BdT sont générés à la main par des agents d'Enedis. Il est donc possible d'envisager des méthodes de TAL pour leur génération automatique. Quant aux réclamations, la génération de réponse est à envisager par la suite.

Remerciements

Je tiens à remercier M. Xavier ROMAN chef du pôle smart-grid et innovation dans laquelle j'ai été affiliée ainsi que M. Romain GERMIGNANI mon tuteur de stage en entreprise pour la confiance qu'il m'a accordé. J'exprime également ma gratitude envers M. Frédéric BECHET, pour son suivi et ses directive.Je remercie également tous les membres de l'équipe se fut un réel plaisir de travailler avec eux.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Benoit Favre, Dilek Hakkani-Tür, and Sebastien Cuendet. 2007. Icsiboost. `https://github.com/benob/icsiboost`.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Antoine Laurent, Nathalie Camelin, and Christian Raymond. 2014. Boosting bonsai trees for efficient features combination : application to speaker role identification. Singapour.

Annexe

Annexe A. Bonzaiboost : Matrice de confusion de la durée

ref/hyp	0	1	2
0	27552	10	214
1	125	51919	143
2	286	262	375898

Table 12: Matrice de confusion sur le train

ref/hyp	0	1	2
0	5718	422	906
1	429	7899	4622
2	714	4206	89187

Table 13: Matrice de confusion sur le test

Annexe B. Détermination du cluster optimal

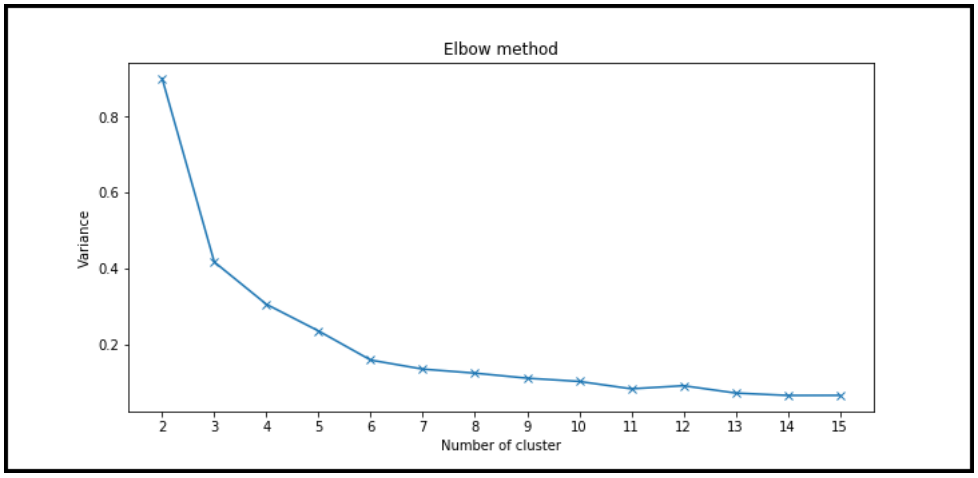

Figure 8: Courbe représentant le nombre de clusters en fonction de la variance. Pour la détermination du cluster optimal avec la méthode d'Elbow.

Annexe C. Résultats sur les BdT comptage

Label	HYP	REF	Correct	Error	Precision	Recall	F-Measure	Error Rate
non	3696	3462	3442	254	93.13	99.42	96.42	7.34
oui	6730	6964	6710	254	99.70	96.35	98.00	3.65
All	10426	10426	10152	274	97.37	97.37	97.37	2.63
Macro F1								97.09
Exact Match								97.37
Multi-Label-Accuracy								97.37
Hamming Loss								1.31

Table 14: Résultat de l'exécution de bonzaiboost pour la prédiction de l'accessibilité du compteur sur le train

Feature	pourcentage
Base comptage	1.0
Particularité du comptage	0.8
Position du comptage	0.6
Accessibilité	3.5
Année de fabrication	17.2
Code postal	27.9
Communication	2.9
Durée estimée	6.7
Objet	31.5
TypeBdt	8.0

Table 15: Importance de chaque feature pour la prédiction de la probabilité d'intervenir

Label	HYP	REF	Correct	Error	Precision	Recall	F-Measure	Error Rate
0	184	179	12	172	6.52	6.70	6.61	96.09
1	3316	3321	3149	172	94.96	94.82	94.89	5.18
All	3500	3500	3161	339	90.31	90.31	90.31	9.69
Macro F1								50.75
Exact Match								90.31
Multi-Label-Accuracy								90.31
Hamming Loss								4.84

Table 16: Résultat de l'exécution de bonzaiboost pour la prédiction de la probabilité d'intervenir sur le dataset test