RAPPORT DE RECHERCHE

Système embarqué de reconnaissance de voix sur un smartphone

SOMMAIRE

[1 Contexte scientifique et technique du projet 7](#_Toc77688576)

[2 Objectifs et performances 7](#_Toc77688577)

[3 Etat de l’art 8](#_Toc77688578)

[3.1 Reconnaissance de la parole 8](#_Toc77688579)

[3.1.1 Paramétrisation 9](#_Toc77688580)

[3.1.2 Le modèle de langage statique 10](#_Toc77688581)

[3.1.3 Le modèle acoustique 10](#_Toc77688582)

[3.1.3.1 Modèle de Markov Caché (HMM) 10](#_Toc77688583)

[3.1.3.2 Modèle hybride 11](#_Toc77688584)

[3.1.4 Décodage avec l’algorithme de Viterbi 11](#_Toc77688585)

[3.2 Reconnaissance de la parole non-native 12](#_Toc77688586)

[3.2.1 La parole non native et son impact sur les performances 12](#_Toc77688587)

[3.2.2 Extraction des règles de confusion phonétique 13](#_Toc77688588)

[3.2.2.1 Modification des HMM en fonction des règles de confusion 14](#_Toc77688589)

[3.2.3 Contraintes graphémiques 15](#_Toc77688590)

[3.2.3.1 Système HMM discret 15](#_Toc77688591)

[3.2.3.2 Utilisation des contraintes graphémiques 15](#_Toc77688592)

[3.2.4 RAP performance 16](#_Toc77688594)

[3.2.4.1 WER: Word Error Rate 16](#_Toc77688595)

[3.2.4.2 CER: le taux d’erreurs sur les caractères 16](#_Toc77688596)

[3.3 Text mining 16](#_Toc77688597)

[3.3.1 Le prétraitement des données 17](#_Toc77688598)

[3.3.1.1 La tokenisation 17](#_Toc77688599)

[3.3.1.2 Supprimer les mots d’arrêt des textes 17](#_Toc77688600)

[3.3.1.3 La lemmatisation/ Stemming 17](#_Toc77688601)

[3.3.1.4 La suppression des bruits 18](#_Toc77688602)

[3.3.2 Le Word embedding 18](#_Toc77688603)

[3.3.2.1 Word to vector (Word2Vec) 18](#_Toc77688604)

[3.3.2.2 Modèle Global Vectors for Word Representation (GloVe) 19](#_Toc77688605)

[3.3.2.3 Réseaux de neurones récurent 19](#_Toc77688606)

[3.3.2.4 Modèle de FastText 21](#_Toc77688607)

[3.3.2.5 Modèle Term frequency/reverse document frequency (TF-IDF) 21](#_Toc77688608)

[3.3.3 Algorithme de classification 22](#_Toc77688609)

[3.3.3.1 Classificateur Naïve Bayes 22](#_Toc77688610)

[3.3.3.2 Support Vector machine (SVM) 23](#_Toc77688611)

[3.3.3.3 Les K plus proche voisin 24](#_Toc77688612)

[3.3.3.4 La forêt aléatoire (Random Forest classifier) 24](#_Toc77688613)

[3.3.4 Evaluation des performances (score) 25](#_Toc77688614)

[3.3.4.1 La matrice de confusion 25](#_Toc77688615)

[3.3.4.2 Le rappel 26](#_Toc77688616)

[3.3.4.3 La F-mesure 26](#_Toc77688617)

[3.4 Le Chatbot (assistant virtuel) 26](#_Toc77688618)

[3.4.1 Le Word embedding 26](#_Toc77688619)

[3.4.2 Identification de l’intention 26](#_Toc77688620)

[3.4.2.1 Approche bayésienne 27](#_Toc77688621)

[3.4.2.2 Approche standard (non-bayésienne) 27](#_Toc77688622)

[3.4.3 Génération de réponses 28](#_Toc77688623)

[3.4.4 Evaluation des performances 29](#_Toc77688624)

[3.5 Benchmark des solutions existantes 29](#_Toc77688625)

[3.5.1 Technique de word embedding 29](#_Toc77688626)

[3.5.2 Technique de classification de texte 30](#_Toc77688627)

[3.5.3 Identification de l’intention (Extraction de l’information) 30](#_Toc77688628)

[4 Incertitudes et difficultés techniques 30](#_Toc77688629)

[5 Sensibilité à l’accent non-natif 30](#_Toc77688630)

[5.1 Le système de reconnaissance en offline 30](#_Toc77688631)

[5.2 Passer de l’audio à l’extraction d’information 31](#_Toc77688632)

[5.3 La rapidité du système RAP 31](#_Toc77688633)

[5.4 L’ensemble des données 31](#_Toc77688634)

[5.5 La résistance aux bruits 31](#_Toc77688635)

[6 Présentation des travaux de R&D 31](#_Toc77688636)

[6.1 Démarche de travail 31](#_Toc77688637)

[6.2 Discussion de l’ancienne version 32](#_Toc77688638)

[6.3 Description des travaux réalisés pour la reconnaissance vocal 32](#_Toc77688639)

[6.3.1 Benchmark des solutions existantes de reconnaissance vocale 32](#_Toc77688640)

[6.3.2 Mozila DeepSpeech 33](#_Toc77688641)

[6.3.3 Langues et dataset 35](#_Toc77688642)

[6.3.3.1 Le dataset Common Voice de Mozilla 35](#_Toc77688643)

[6.3.3.2 Choix de la taille du vocabulaire du corpus 37](#_Toc77688644)

[6.3.4 Expérimentation 37](#_Toc77688645)

[6.3.4.1 Reconnaissance de parole sur un corpus de large vocabulaire LVCSR 37](#_Toc77688646)

[6.3.4.2 Reconnaissance de la parole sur un corpus de faible vocabulaire (SVCSR) 37](#_Toc77688647)

[6.3.4.3 Préparation des données 38](#_Toc77688648)

[6.3.4.4 Création d’un modèle de langage 39](#_Toc77688649)

[6.3.4.5 Relancer l’ancienne version (version 2020) 39](#_Toc77688650)

[6.3.5 Conclusion global de la reconnaissance vocal: 48](#_Toc77688654)

[6.4 Description des travaux réalisés pour text mining 48](#_Toc77688655)

[6.4.1 Text mining: classification automatique de texte 48](#_Toc77688656)

[6.4.2 Importer et charger le fichier de donnée 48](#_Toc77688657)

[6.4.3 Le prétraitement des données 49](#_Toc77688658)

[6.4.4 Création des données de formation et de test 50](#_Toc77688659)

[6.4.5 Le word embedding 51](#_Toc77688660)

[6.4.5.1 TF-IDF 51](#_Toc77688661)

[6.4.6 Création et évaluation de modèles 51](#_Toc77688662)

[6.4.7 Réglage de l’hyper paramètre du modèle: 52](#_Toc77688663)

[6.4.8 Visualisation des résultats: 52](#_Toc77688664)

[6.4.8.1 Matrice de confusion de naïve bayésien: 53](#_Toc77688666)

[6.4.8.2 Classificateur support à vecteur machine 55](#_Toc77688669)

[6.4.8.3 Classificateur de forêt aléatoire 56](#_Toc77688671)

[6.4.8.4 Classificateur régression logistique 57](#_Toc77688672)

[6.4.9 Comparaison de différents classificateurs: 58](#_Toc77688673)

[6.4.10 Conclusion du text mining: 58](#_Toc77688674)

[6.5 Travaux realisé pour le Chatbot 58](#_Toc77688675)

[6.5.1 Chatbot 58](#_Toc77688676)

[6.5.2 Chatbot basé sur la récupération 59](#_Toc77688677)

[6.5.3 Importer et charger le fichier de données 59](#_Toc77688678)

[6.5.4 Le prétraitement des données 59](#_Toc77688679)

[6.5.5 Création des données de formation et de test 60](#_Toc77688680)

[6.5.6 Construire le modèle d’entrainement pour la classification des intentions 61](#_Toc77688681)

[6.5.7 Compilation du modèle 61](#_Toc77688682)

[6.5.8 Résultat obtenue pour l’ensemble d’entrainement 62](#_Toc77688684)

[6.5.9 Evaluation du modèle 62](#_Toc77688687)

[6.5.10 Exemple de classification 62](#_Toc77688690)

[6.5.11 Génération de réponse 63](#_Toc77688691)

[6.5.12 Visualisation du résultat 63](#_Toc77688692)

[6.5.13 Conclusion global du Chatbot 63](#_Toc77688694)

[7 conclusion 64](#_Toc77688695)

[8 PROGRES SCIENTIFIQUES ET/OU TECHNIQUES 64](#_Toc77688696)

[8.1 Valeur ajoutée 64](#_Toc77688697)

[8.2 Perspectives 65](#_Toc77688698)

[9 BIBLIOGRAPHIE 66](#_Toc77688699)

[10 Annexe 69](#_Toc77688700)

[Annexe1: Résultats du test de la RAP 69](#_Toc77688701)

[Annexe 2: Code calcul du WER et CER 69](#_Toc77688702)

[Annexe 3: Code distance de Levenshtein 70](#_Toc77688703)

Figure

[Figure 1: la procédure générale pour la reconnaissance de la parole 7](#_Toc77688539)

[Figure 2: Architecture d’un système automatique de reconnaissance vocale [2] 7](file:///C:\Users\SADDEDINE_H\Desktop\SMA_IDF_23%20Système%20embarqué%20de%20Reconnaissance%20voix%20sur%20un%20smartphone_Vfinale1%20(Corrigé%20VS).docx#_Toc77688540)

[Figure 3: Exemples d’un HMM avec 6 états d’observations 10](#_Toc77688541)

[Figure 4: Deux styles d’HMM, phonème et mot 10](#_Toc77688542)

[Figure 5: prononciation du phonème **∂** en anglais 11](#_Toc77688543)

[Figure 6: l’alignement phonétique en utilisant le système de RAP de C du mot approach](#_Toc77688544)

[Figure 7:la reconnaissance phonétique du mot approach 12](#_Toc77688545)

[Figure 8:HMM modifié pour le phonème anglais [Tʃ] 14](#_Toc77688546)

[Figure 9: vue d'ensemble du processus de préparation des textes 16](#_Toc77688547)

[Figure 10: L’architecture continue du sac de mots (CBOW) prédit le mot actuel en fonction du contexte, et le Skip-gram prédit le mot voisin basé sur le mot actuel 18](#_Toc77688548)

[Figure 11: Exemple de LSTM à trois couches 19](#_Toc77688549)

[Figure 12: Exemple de GRU 20](#_Toc77688550)

[Figure 13: Méthode de classification de texte 21](#_Toc77688551)

[Figure 14: Vecteur à support machine 22](#_Toc77688552)

[Figure 15: Architecture d’une foret aléatoire 23](#_Toc77688553)

[Figure 16:: Architecture global d'un Chatbot 25](#_Toc77688554)

[Figure 17:Architecture d’un multi perceptron 27](#_Toc77688555)

[Figure 18: Un exemple basé sur la récupération de réponse 27](#_Toc77688556)

[Figure 19: exemple de génération de réponse basé sur les techniques génératives 27](#_Toc77688557)

[Figure 20: Architecture RNN 33](#_Toc77688558)

[Figure 21: Répartition des langues du Dataset 34](#_Toc77688559)

[Figure 22: les accents en français 35](#_Toc77688560)

[Figure 23: fichiers CSV 37](#_Toc77688561)

[Figure 24: Evolution de WER et CER en fonction de nombre d'epoch 45](#_Toc77688562)

[Figure 25: Exemple de résultat absurde 46](#_Toc77688563)

[Figure 26: Visualisation des résultats du classificateur Naïve bayésien 52](#_Toc77688564)

[Figure 27: Visualisation des résultats du support à vecteur machine 54](#_Toc77688565)

[Figure 28: Visualisation des résultats du classificateur de forêt aléatoire 55](#_Toc77688566)

[Figure 29: Visualisation des résultats du classificateur régression logistique 56](#_Toc77688567)

[Figure 30: Histogramme présentant la moyenne Macro F1-score des différents classificateurs](#_Toc77688568)

[Figure 31: fichier de question/réponse 58](#_Toc77688569)

[Figure 32: Exemple de prétraitement des données 59](#_Toc77688570)

[Figure 33: Résultat Bag Of Word 59](#_Toc77688571)

[Figure 34: Architecture du modèle 60](#_Toc77688572)

[Figure 35: Exemple de classification de l’intention de l’utilisateur 61](#_Toc77688573)

[Figure 36: Generation de réponse 62](#_Toc77688574)

Tableau

[Tableau 1: Association phonème-graphème pour le mot "speech" 15](#_Toc77689009)

[Tableau 2: Exemple donnée fichier TSV 35](#_Toc77689010)

[Tableau 3: Impact de nombre d’epoch sur la précision 45](#_Toc77689011)

[Tableau 4: Impact de la taille du lot sur la précision 46](#_Toc77689012)

[Tableau 5: Descriptif des hyper-paramètres de décodeur CTC 47](#_Toc77689013)

[Tableau 6: Résultat obtenue avec différents classificateurs 57](#_Toc77689014)

# Contexte scientifique et technique du projet

SEGULA Technologie assure auprès de ses clients l’activité de roulage qui représente tous les tests liés aux essais des véhicules. Les rouleurs sont les personnes réalisant ces tests. Ils sont chargés de remonter les défauts de fabrication, qui doivent être remontés au constructeur afin de pouvoir les analyser par la suite. Pour décrire ces anomalies, les rouleurs complètent leurs rapports sur format papiers. Ces rapports sont un moyen de dialogue entre toutes les équipes de roulage. Ces équipes comportent un nombre important de locuteurs non-natifs.

Cette technique de transmission comprend deux problématiques principales. La première problématique concerne le temps de mobilisation des ressources humaines, dont le temps de saisie du rouleur au cours de la gamme et le temps de conversion du format papier au format digital. La deuxième problématique est qu’il peut y avoir une perte d’informations lors de cette saisie. Cela peut être dû à une mauvaise description de la part du rouleur, à une mauvaise compréhension de la personne qui numérise le rapport, ou simplement à la perte du document.

Une première version de l’outil de digitalisation avec une interface vocale intégrée a été développée durant les travaux précédents : afin de pallier les problématiques citées précédemment. Ce système a pour but de permettre aux rouleurs de dicter sur une tablette ou un téléphone portable les anomalies qu’ils rencontrent, afin de réduire les délais de saisie et de rédiger des comptes rendus complets. Le but de ce projet de recherche est de poursuivre la construction du modèle de reconnaissance vocale afin de le rendre sensible aux accents français non-natifs, et ainsi pouvoir l’utiliser dans des pays non-francophone et mode hors connexion.

# Objectifs et performances

Le but de ce projet est de changer la technique de transmission des anomalies de roulages en élaborant un système de reconnaissance de la parole. L’objectif de ce projet est d’augmenter la robustesse de l’interface vocale dans des conditions réelles de fonctionnement (le bruit ambiant du véhicule, le vocabulaire, la grammaire et les accents des rouleurs).

Cet outil doit comporter une fonctionnalité qui convertit les textes en données. L’objectif est de détecter et d’extraire des informations pertinentes à partir de ces comptes rendus.

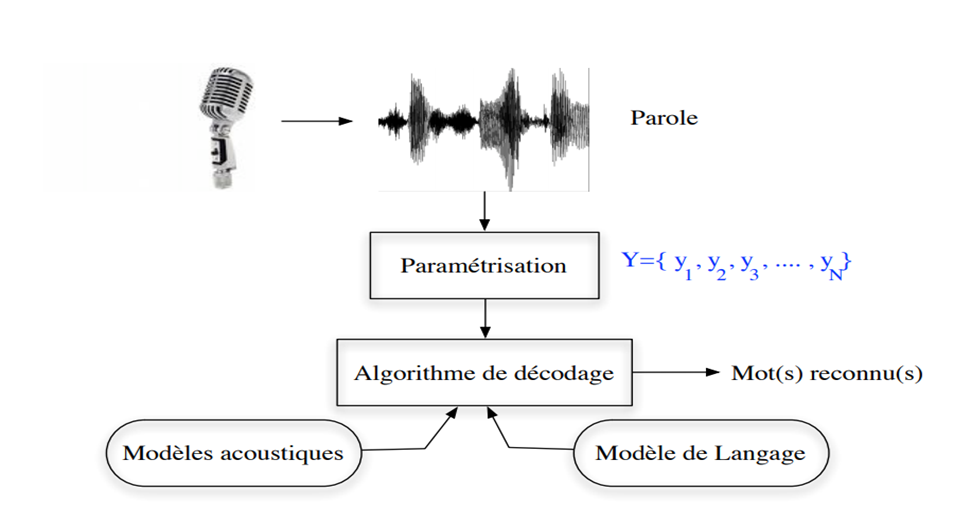
De plus, il est important que l’outil soit indépendant d’une connexion internet car cette connexion est intermittente sur les pistes de roulage. C’est pour cela que l’outil de reconnaissance de vocal doit être entièrement embarqué.

Pour mieux interagir avec ces rouleurs, il faut que l’outil possède un assistant virtuel (Chatbot). Après chaque phrase émise par le rouleur, cet assistant proposera une liste permettant de mieux caractériser le défaut observé afin de rédiger des comptes rendus plus complets et plus précis. Il pourra également utile pour donner des instructions si le rouleur a besoin d’aide (Numéro à contacter en cas de panne …).

# Etat de l’art

## Reconnaissance de la parole

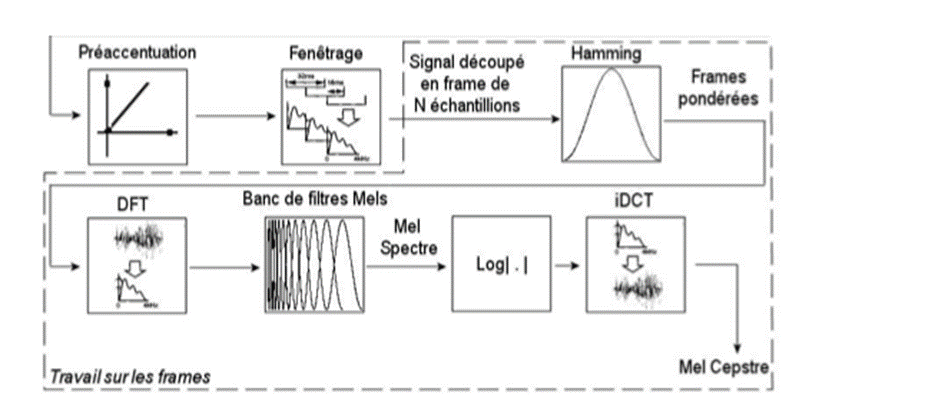
La reconnaissance de la parole basée sur le système de reconnaissance automatique de la parole (SAP) se fait avec le modèle HMM (hidden model Markov). Il se construit en deux étapes: la première est la paramétrisation du signal dans le but d’extraire les informations pertinentes. La seconde est le décodage du signal avec l’algorithme de Viterbi (modèle acoustique, le lexique et la modèle de langage)

Figure 1 : la procédure générale pour la reconnaissance de la parole

### **Paramétrisation**

La paramétrisation la plus utilisée pour la RAP est MFCC permet de convertir le signal acoustique du domaine temporel en un signal spectral. L'échelle Mel, utilisée dans la paramétrisation MFCC, prend en compte la perception des changements de fréquences sonores par l'oreille humaine.

En effet, l'oreille humaine est plus sensible aux changements de fréquences sonores dans les basses fréquences que dans les hautes fréquences. Comme les coefficients des MFCC sont très influencés par les distorsions des bruits, cela implique plusieurs traitements du signal audio:

 Figure 2 : Paramétrisation MFFC

Le rôle de la préaccentuation est d’augmenter les fréquences en vue d’en faciliter la transmission.

Le fenêtrage permet simplement le découper ce signal en petites trames permettant de mieux analyser le signal audio. La paramétrisation est effectuée sur des trames successives de signal de courte durée (pour lesquelles le signal peut être considéré comme étant quasi stationnaire, typiquement 25 ms). Les trames successives se recouvrent et le décalage entre deux trames successives est en moyenne de 10 ms.

En coupant le signal, on introduit des hautes fréquence parasites. Lors de la paramétrisation MFFC, il est appliqué une fenêtre de Hamming [Cf. figure 2] pour filtrer le début et la fin du signal. Le passage du domaine temporel vers le domaine spectral par la transformée de Fourier discrète permet de calculer l’estimation du spectre de chaque trame. Un filtrage est ensuite appliqué sur le spectre, en utilisant les bancs de filtres Mels.

L’échelle de Mel modélise la non-linéarité de la perception humaine de l’audio au niveau des fréquences. mel(f) = 2595 log10(1 + f / 700) (1)

Enfin, pour revenir dans un pseudo domaine temporel, on applique la transformée de Fourrier en cosinus discrète inverse(DFT). A partir de ce signal traité, on récupère son spectre qui est alors prêt à être analysé.

### **Le modèle de langage statique**

Le modèle de langage définit la suite des mots que le système peut reconnaître.

Ce modèle consiste pour une séquence de mots suivante M = m1, m2, …, mN

à calculer sa probabilité P(M) =∏ P(mi|m1……..m(i-1)=∏ p(mi|hi) (2)

Chaque fois que la séquence de mots hi s’enrichit, une estimation des valeurs des probabilités conditionnelles P(mi|hi) devient de plus en plus difficile. Le modèle le plus utilisé pour résoudre ce problème est le modèle de n-gram. Le modèle de n-gram se base sur l’approximation que la probabilité d’un mot dépend que de ceux qui le précèdent (qu’à partir d’une séquence de donnée il est possible d’obtenir la fonction vraisemblable du mot suivant). Cette modélisation correspond à un modèle de Markov caché d’ordre n ou seules les n dernières observations sont utilisées pour la prédiction de la lettre suivante:

P(W) = P (w1, w, .., wm−1,wm) = P(w1) · P(w2|w1) ·..· P (wm|w1, w2,..., wm−1) (3)

Les probabilités d'apparition des mots sont estimées au maximum de vraisemblance selon l’équation.

P( ω\_n |ω\_1,ω\_2,...,ω\_(n-1)) ==(C( ω\_1,ω\_2,...,ω\_n))/(C( ω\_1,ω\_2,...,ω\_(n-1)))

Où C (x) est le nombre d'occurrence de la suite de mots x dans le corpus textuel d'apprentissage.

### **Le modèle acoustique**

#### Modèle de Markov Caché (HMM)

Un HMM modélise un processus markovien de paramètres inconnus et dont les états sont inconnus. Un processus markovien est un système possédant un nombre finis d’états et émettant des observations au cours du temps. Chaque état émet des « observations » selon une probabilité d’émission qui sont observables. C’est sur la séquence d’observations que l’on travaille et non la séquence d’états comme sur un modèle de Markov. Ci-dessous, une représentation d’un HMM avec six états:

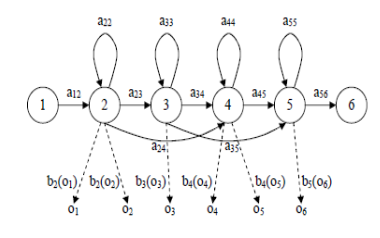


Figure 3 : Exemples d’un HMM avec 6 états d’observations

Avec: la probabilité de transition de l’état à l’état avec, N étant le nombre d’états, et

la probabilité d’observation, étant l’indice émis par l’état actuel, avec, m est le nombre de symboles d’observation, et

l’observation

On peut définir un HMM niveau du phonème ou au niveau du mot:

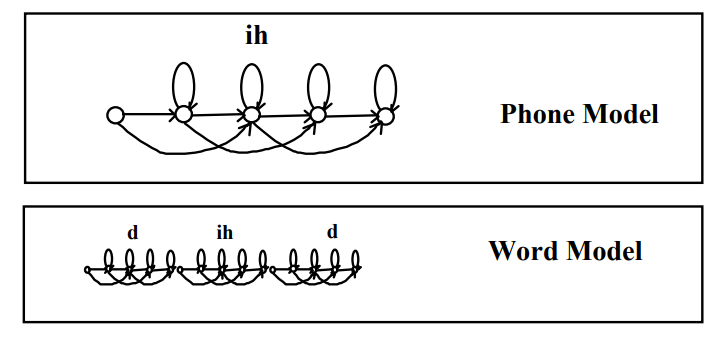


Figure 4 : Deux styles d’HMM, phonème et mot

Historiquement, un HMM était couplé avec un mélange de gaussiennes (Gaussian Mixture Model ou GMM). Il existe aussi des variantes telles que HMM2 [3], qui est un HMM d’ordre 2 ou bien des méthodes d’adaptation du modèle acoustiques comme les méthodes Maximum a Posteriori (MAP) [1], [3] et Maximum Likelihood Linear Regression(MLLR) [1].

Le couple HMM-GMM était utilisé jusque dans les années 2000, où les progrès dans les réseaux de neurones et l’apparition de modèles hybrides ont rendu son usage obsolète.

#### Modèle hybride

Les méthodes les plus récentes de modélisation acoustique proposent des réseaux de neurones profonds dépendants du contexte (context dependent deep neural network, CD-DNN) à la place des distributions multi-gaussiennes des probabilités d’émission, et fournissent des améliorations significatives par rapport aux modèles HMM-GMM. Leur succès provient de l’utilisation de plusieurs couches cachées (généralement entre 5 et 7), chacune ayant environ 2048 unités avec un sigmoïde non-linéaire. La couche de sortie a une non-linéarité softmax et un nombre d’unités de sortie égal au nombre d’états HMM. De plus, l’architecture de calcul utilisant des processeurs graphiques (GPU) permet de paralléliser efficacement l’apprentissage et le décodage de la parole.

Les extensions récentes des modèles DNN incluent les réseaux de neurones convolutifs et les réseaux de neurones récurrents.

### **Décodage avec l’algorithme de Viterbi**

Une fois les modèles créés, il faut maintenant passer à la phase de décodage. Cette phase consiste à retrouver la suite de mot W^\* qui maximise la probabilité P(W|X), X étant le signal audio en entrée.

L’équation ci-dessous peut être simplifiée:

Avec O la séquence de vecteurs d’observations acoustiques.

Deux solutions existent :

-Calculer P(O│W) pour chaque état possible et ne retenir que la meilleure. En construisant un arbre, à chaque instant t, une couche de nœuds est ajoutée à l’arbre ou chaque nœud représente un état particulier et la probabilité de se retrouver dans cet état à l’instant t. cette technique n’est pas fiable voir inapplicable car le nombre d’hypothèse est trop grand.

-Le décodage avec l’algorithme de Viterbi propose de simplifier l’arbre au fur et à mesure de sa construction. Lors de son déroulement, plusieurs hypothèses peuvent se retrouver dans un même état à l’instant t. l’algorithme de Viterbi fait en sorte de pas dérouler les hypothèses de faible probabilité car elles ne peuvent pas être utilisées pour décrire le message de plus probable.

## Reconnaissance de la parole non-native

### La parole non native et son impact sur les performances

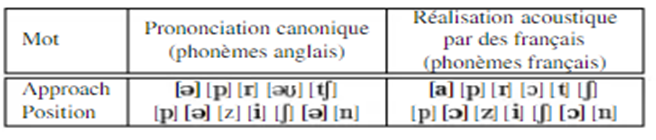
On dit qu’un locuteur est non natif quand il parle une langue qui n’est pas sa langue maternelle. La parole non-native diffère de la parole native (canonique) par le fait qu'elle peut contenir des erreurs de prononciation. Les locuteurs non-natifs ont tendance à prononcer les phonèmes d'une manière similaire à leur langue maternelle. Les performances de la RAP se dégradent face à ces erreurs de prononciations.

Figure 5 : prononciation du phonème ∂ en anglais

Le type d’erreurs illustré dans la figure 4 dépendent de la langue maternelle des locuteurs et leurs capacités à reproduire de la langue cible. Il s'agit de modifier un système de RAP d'une langue cible en utilisant des approches d'adaptation à l'accent non-natif. Pour chaque phrase du corpus d’adaptation:

- un alignement forcé du signal audio avec la suite de modèles acoustiques de la langue parlée correspondant à la transcription canonique de la phrase.

- une reconnaissance phonétique du signal audio avec des modèles acoustiques de la langue maternelle.

L’avantage de l’utilisation de la confusion minimise le surcoût de puissance de calcul pour le nouveau SRAP.

### Extraction des règles de confusion phonétique

La première règle est un alignement phonétique qui consiste à utiliser la transcription de la parole prononcée afin de déterminer les intervalles de temps d'occurrence des phonèmes.

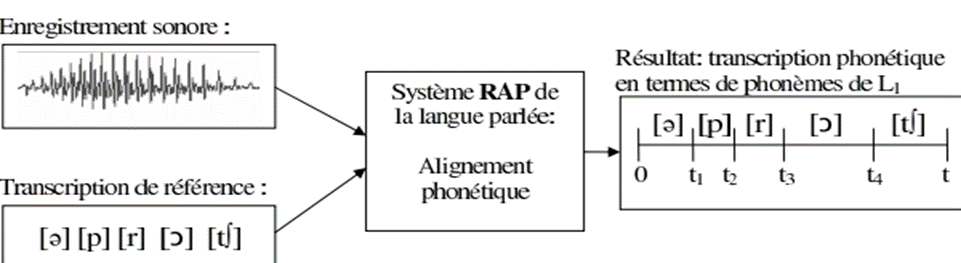


Figure 6 : l’alignement phonétique en utilisant le système de RAP de C du mot approach

Comme on peut le constater sur cette figure, la transcription de l'enregistrement sonore est fournie au système de RAP.

La seconde règle est que le système RAP est mis à profit pour effectuer une reconnaissance phonétique sur le corpus. La transcription de l'enregistrement n'est pas fournie au système de RAP. Il s'agit ici de déterminer quels phonèmes de la langue Oi ont été prononcés et les temps d'occurrence de chacun.

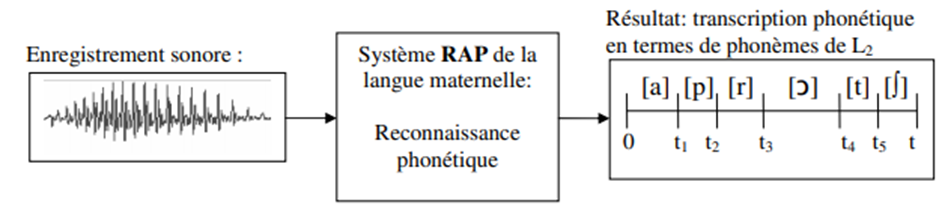
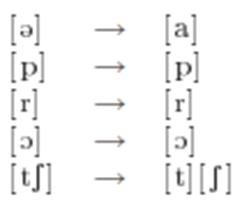


Figure 7 : la reconnaissance phonétique du mot approach

Une fois que ces deux règles sont appliquées sur tout le corpus de la parole non native, l’étape finale est de comparer ces deux transcriptions à un phonème de référence p. De cette manière, les erreurs d'alignement et de reconnaissance phonétiques auront une influence réduite sur l'extraction d'associations

À chaque occurrence d'un phonème de la transcription de référence p, une association est extraite comme décrit plus haut. Pour chaque phrase du corpus de parole non-native, un ensemble d'associations phonétiques A, avec leurs nombres d'apparitions n(a), ∀a ∈ A est extrait en comparant la transcription de référence et celle reconnue.

Ensuite, on sélectionne les associations les plus pertinentes. Elles sont retenues si elles

Vérifient cette équation n(ap) ≥ α max a∈Ap {n(a)} (6)

Ap est l'ensemble des associations pour le phonème de référence p, α un facteur vérifiant 0 < α < 1

#### Modification des HMM en fonction des règles de confusion

Les modèles de HMM sont modifiés à l’aide de règles de confusions décrite ci-dessus. Pour un phonème p de la langue parlé un chemin alternatif est ajouté au HMM.

Pour chaque règle rp ∈ Rp (Rp l’ensemble des règles sélectionnées pour le phonème p) donnera lieu à un chemin d'états alternatifs dans le modèle de p qui compose de la concaténation des modèles des phonèmes présents dans la partie droite de rp.

Le modèle HMM aura:

-Un chemin correspondant au modèle canonique du phonème p (sans aucune modification).

-Un chemin correspondant à chaque règle de Rp (Rp l’ensemble des règles sélectionner pour le phonème p)

L’utilisation de cette procédure se fait comme suit:

[Tʃ]=======> [t][ʃ] P([t ʃ]=======> [t][ ʃ])=0.6

[Tʃ]=======> [ʃ] P([t ʃ]=======> [ ʃ] )=0.4

Le modèle contient trois chemins disjoints et distincts représentant respectivement la prononciation canonique du phonème [[Tʃ], la première prononciation non-native [ʃ] et la seconde prononciation non-native [t][ʃ]. À la phase de reconnaissance, une seule prononciation pourra être choisie (associée à un seul chemin du nouveau modèle HMM).

Le facteur β ∈]0,1[est une pondération entre la prononciation canonique et les prononciations non-natives. De même, la probabilité liant cet état non émetteur à chaque chemin ajouté (pour une règle r ∈ RK) est calculée comme suit:

Pour P([t][ʃ]): P’([t ʃ])=(1- β)\* P([t ][ʃ])

Pour P([ʃ]): P’([ ʃ])=(1- β)\* P([ʃ])

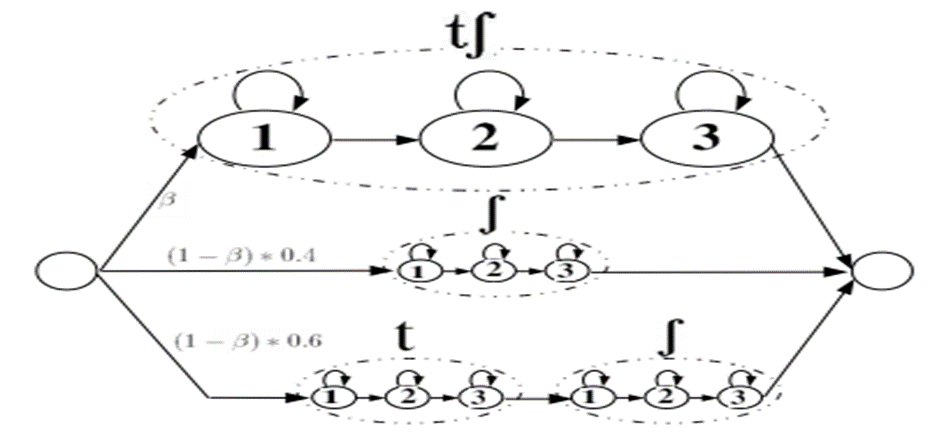


Figure 8 : HMM modifié pour le phonème anglais [Tʃ]

### Contraintes graphémiques

Les erreurs réalisées par les locuteurs sont souvent liées à la graphie des mots. Les locuteurs prononcent les phonèmes de la langue cible, en l'occurrence l'anglais, d'une manière différente selon les mots. Plus précisément, la prononciation d'un phonème dépend des lettres (graphèmes) correspondant. Pour pallier ce problème, il existe une méthode d’alignement phonème-graphème. Son but est d’associer les phonèmes aux graphèmes. Le système HMM discret peut être utilisé pour effectuer l’alignement phonème-graphème sur le dictionnaire du SRAP cible, via l’algorithme de Viterbi.

#### Système HMM discret

L’alignement phonème-graphème est extrait d’une manière automatique à partir d’un grand dictionnaire phonétique. Dans ce système, les modèles HMM représentent les phonèmes, les observations discrètes représentent les caractères et le dictionnaire phonétique représente le corpus d’apprentissage.

Les modèles HMM discrets initiaux ont une probabilité d’émission uniforme pour tous les symboles discrets (correspondant chacun à un graphème). Enfin, pour chaque mot du dictionnaire d’apprentissage, un fichier de données discrètes correspondant à la suite de symboles discrets (graphèmes du mot) est créé.

Un alignement phonétique est effectué sur le dictionnaire d’apprentissage afin de déterminer les associations entre les phonèmes et les graphèmes. Seules les associations les plus représentées sont retenues. Une association A phonème-graphème elle est retenue si elle vérifié cela: n(αL)≥ γ∑\_(x∈A)〖n(x)〗 (7)

Où A est l'ensemble d'associations relatives au phonème L, n(aL) le nombre d'apparitions de l'association a, et γ est un facteur

#### Utilisation des contraintes graphémiques

Le modèle de HMM discrets est obligatoire afin de procéder à un alignement forcé sur le lexique de la langue cible. La méthode simple pour appliquer les contraintes graphémiques au SRAP cible est de remplacer les phonèmes dans la prononciation des mots du lexique par le couple (phonèmes, graphème) pour chacun des mots du dictionnaire (SRAP cible). La prononciation d’un mot n’est plus une suite de phonème, mais une suite de couple (phonèmes, graphèmes). L’exemple suivant illustre ceci pour le mot anglais “speech”

#### 

Tableau 1 : Association phonème-graphème pour le mot "speech"

À cet effet, un alignement forcé sur le dictionnaire du SARP cible est effectué en utilisant le système HMM discret. Les phonèmes, dont les association (phonème-graphème) ne sont pas retenues, restent sans contrainte graphémiques dans la prononciation du mot.

### RAP performance

#### WER : Word Error Rate

Les performances sont mesurées sur le taux d’erreurs en mots avec WER. Si WER=0 la reconnaissance est bonne. Et si WER>=1 cela implique qu’y a beaucoup d’erreurs. L’équation de calcule de WER est donnée de la manière suivante (la distance de Levenshtein):

S∶ nombre de substitutions de mots

D∶ nombre de suppressions

I∶ nombre d’insertions

N∶ nombre de mots dans la référence

#### CER : le taux d’erreurs sur les caractères

Les performances sont mesurées par rapport au taux d’erreur sur les paires attribut/valeur. Un concept détecté est considéré comme correct uniquement si l’attribut du concept et sa valeur normalisée sont corrects d’après la référence. L’équation de calcule de CER est donnée de la manière suivante:

I∶ nombre d’insertions

S∶ nombre de substitutions de mots

O:  le nombre de concepts omis

R: nombre de mots dans la référence

## Text mining

On met en œuvre des techniques de texte mining sur les transcriptions des anomalies enregistrées via interface vocale par les rouleurs. L’objectif est d’extraire les informations pertinentes.

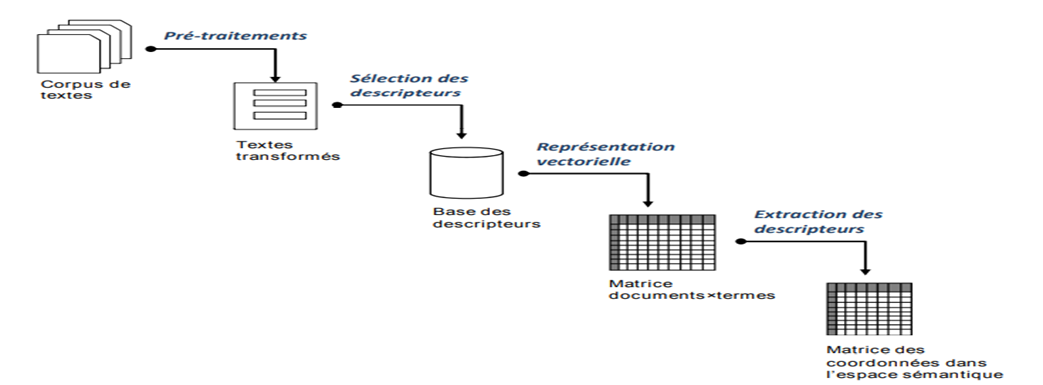
La chaîne de traitement du text Mining est la suivante: la lemmatisation/stemming, l’extraction de concept métier, segmentation et le classement. Ces traitements permettent de classer ces transcriptions selon différentes thématiques.

Figure 9 : vue d'ensemble du processus de préparation des textes

### Le prétraitement des données

#### La tokenisation

Les textes reconnus par le système de RAP ne peuvent pas être analysés directement. La première étape est de décomposer le texte en mots séparés, pour ensuite les utilisés pour convertir chaque mot en un nombre réel. La tokenisation est le processus qui décompose le flux de texte en mot. L’objectif principal de cette étape est d’extraire des mots individuels dans une phrase.

#### Supprimer les mots d’arrêt des textes

Les mots d’arrêt sont des mots courants qui apparaîtront probablement dans n’importe quel texte. Ils ne sont pas révélateurs au sujet des données, alors on les exclut de l’analyse du flux de texte. Par exemple: « ‘l’argent ou le plomb est très bien pour moi -> argent, plomb, très bien’ ».

#### La lemmatisation/ Stemming

Lors de la reconnaissance vocale, les rouleurs peuvent parler de différentes manière (au présent, au futur, au passé…). Alors cette technique vise à réduire les différentes formes d’un mot à une racine commune (ils ne prennent pas en compte la flexion des mots). Cette technique permet de réduire la taille des fichiers d’index.

Le but est de continuer à réduire le nombre de mots considérés. Pour cela, il est possible de rassembler les mots faisant partie de la même famille ou possédant la même racine. C’est le but de la lemmatisation et du stemming. En effet, la lemmatisation permet d’extraire les formes canoniques des mots. Par exemple : [mange], [mangera], [mangeaient] deviennent [manger]. Le stemming utilise la racine des mots plus la dérivation (préfixe, suffixe). Par exemple : [production], [productivité] deviennent [product]

#### La suppression des bruits

Les documents texte contiennent généralement des caractères comme des ponctuations ou des caractères spéciaux et ils ne sont pas nécessaires pour l’exploration de texte ou de classification. Bien que la ponctuation soit essentielle pour comprendre le sens de la phrase, elle peut affecter négativement les algorithmes de classification.

### Le Word embedding

Cette étape consiste dans un premier temps à convertir le texte reconnu par le système RAP, en un vecteur de nombres réels, parce que le langage machine comprend uniquement les chiffres. Donc le Word embedding désigne les techniques qui visent à représenter les phrases d’un texte par un vecteur de nombres réels. Il est capable de capturer le contexte d’un mot dans un document, la similitude sémantique et syntaxique, la relation avec d’autres mots, etc. Quelques techniques de Word embedding sont décrite brièvement ci-dessous.

#### Word to vector (Word2Vec)

C’est une technique qui repose sur les réseaux de neurone. Cette technique prend comme entrée le texte reconnu par le système de RAP et produit des vecteurs de nombres réels. Chaque mot unique dans [le corpus](https://en.wikipedia.org/wiki/Corpus_linguistics) est attribué à un vecteur correspondant dans l’espace. [Les vecteurs de mots](https://en.wikipedia.org/wiki/Word_vectors) sont placés dans l’espace vectoriel de telle sorte que les mots qui partagent des contextes communs dans le corpus soient situés à proximité. Il convertit les mots en probabilités softmax (un nombre compris entre 0 et 1) d’une dimension donnée.

Cela peut être fait en utilisant deux approches différentes. À partir d’un seul mot w pour prédire son contexte C (Skip-gram) ou à partir du contexte pour prédire un mot (Sac continu de mots). Pour créer un vecteur de haute dimension pour chaque mot. L’objectif est de maximiser la probabilité conditionnelle:

Ou T se réfère au texte

Θ est le paramètre de p (c | w; θ)

C(w) est l’ensemble des contextes de mot w

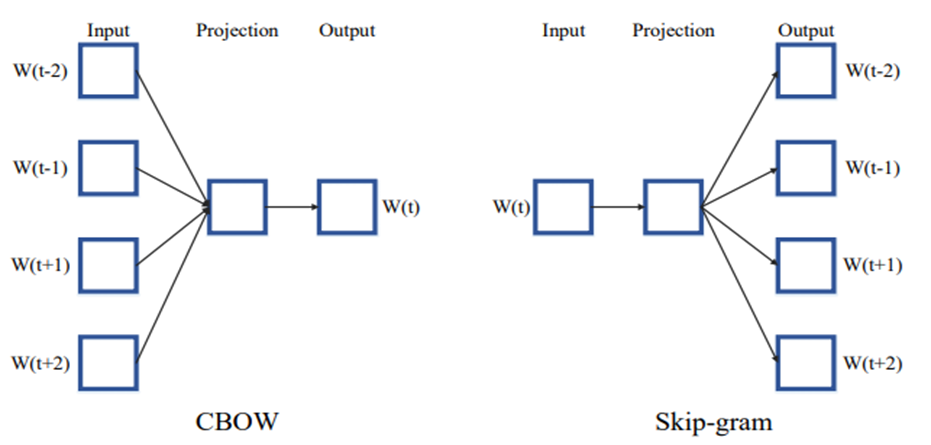


Figure 10 : L’architecture continue du sac de mots (CBOW) prédit le mot actuel en fonction du contexte, et le Skip-gram prédit le mot voisin basé sur le mot actuel

Le CBOW vise à prédire un mot en fonction de son contexte, c'est-à-dire selon les mots qui en sont proches dans le texte. Le skip-gram a une architecture symétrique visant à prédire les mots du contexte en fonction d’un mot en entrée. Le modèle CBOW apprend plus rapidement, mais le modèle skip-gram donne généralement de meilleurs résultats.

#### Modèle Global Vectors for Word Representation (GloVe)

GloVe est un autre modèle de Word embedding similaire à Word2Vec. Il est basé sur des techniques de factorisation matricielle sur la matrice mot-contexte. GloVe construit d’abord une grande matrice d’informations de cooccurrence (mots x contexte), c’est-à-dire pour chaque mot (les lignes), on compte la fréquence à laquelle on voit ce mot dans un certain contexte (les colonnes).

Le nombre de contextes est bien sûr important puisqu’il est essentiellement de taille combinatoire. En factorisant cette matrice, on obtient une matrice de dimensions inférieures (mots x caractéristiques), où chaque ligne donne une représentation vectorielle pour chaque mot. En général, cela se fait en minimisant une perte de reconstruction.

Cette perte tente de trouver les représentations de dimensions inférieures qui peuvent expliquer la majeure partie de la variance dans les données de haute dimension.

#### Réseaux de neurones récurent

Les RNNs sont un type de réseaux de neurone avec des boucles spécialement adapté séquence de données. Plus formellement, ils contiennent un état interne généralement noté Ct pour l’état à un instant t, qui est réinjecté comme entrée pour l’état prochaine(t+1). Cependant, le principal problème avec l’implémentation de ces réseaux réside dans le fait qu’ils souffrent du gradient explosif. Pour pallier à ce problème, plusieurs variant de RNN ont été développés.

##### Unité de mémoire à long-court terme (LSTM)

Les composants principaux de LSTM sont les portes d’oubli et d’entrée, l’état de la cellule et la porte de sortie. Soit l’entrée et l’état caché produit par LSTM. La relation entre x et h est donnée par l’équation suivante:

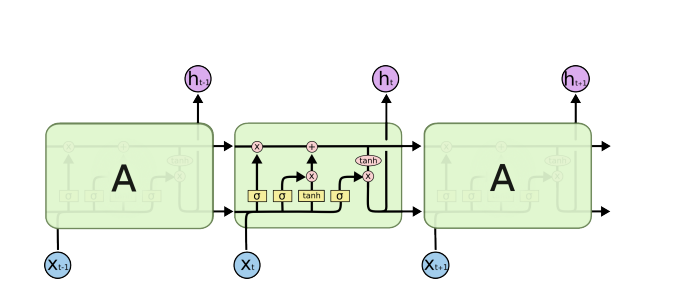


Figure 11 : Exemple de LSTM à trois couches

=σ()

= σ()

: matrice de poids de la porte d’oubli (foret gate)

: matrice de poids de la porte d’entrée associé aux couches cachés

: matrice de poids de la porte d’oubli

État de la cellule précédente

Matrice de poids d’entrée de la porte de sortie

Matrice de poids de sortie de la porte de sortie

##### Gate recurrent units

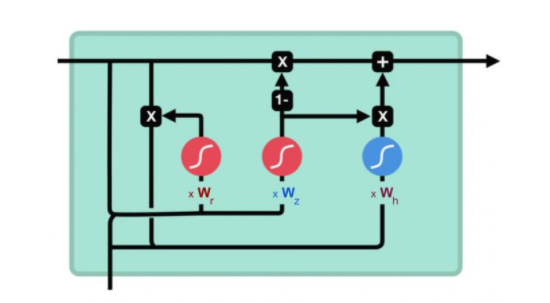
C’est une variante des LSTM avec moins de paramètres à régler et donc moins de temps de formation.

Figure 12 : Exemple de GRU

)

Où z et r représentent respectivement les portes de mise à jour et de réinitialisation.

Bien et h représentent respectivement la mémoire intermédiaire et la sortie.

#### Modèle de FastText

FastText est une bibliothèque pour l’apprentissage efficace des représentations de mots et la classification des phrases.  Le modèle est un algorithme d’apprentissage non supervisé pour obtenir des représentations vectorielles pour les mots. Il traite chaque mot comme composé de n-gram de caractère. Ainsi, le vecteur d’un mot est constitué de la somme de ce caractère n-gram.

#### Modèle Term frequency/reverse document frequency (TF-IDF)

Fréquence terme (TF) donne la fréquence du mot à chaque document du texte reconnu par RAP. C’est le rapport entre le nombre de fois que le mot apparaît dans le texte par rapport au nombre total de mots dans tout le texte. Chaque document à sa propre TF. Elle est donnée par l’équation ci-dessous

Fréquence inverse des données (IDF) est utilisée pour calculer le poids des mots rares dans tous le texte. Les mots qui se produisent rarement dans le texte ont un score élevé. Ils sont donnés par l’équation ci-dessous

Le score TF-IDF est trouvé en combinant les deux équations décrite ci-dessous

Le nombre d’occurrence de i à j

Le nombre de documents contenant i dans le corpus

Le nombre total de document

En résumé, le but de Word embedding est de transformer les mots en nombres, que des algorithmes comme l’apprentissage profond peuvent alors traiter, pour formuler une compréhension du langage naturel.

### Algorithme de classification

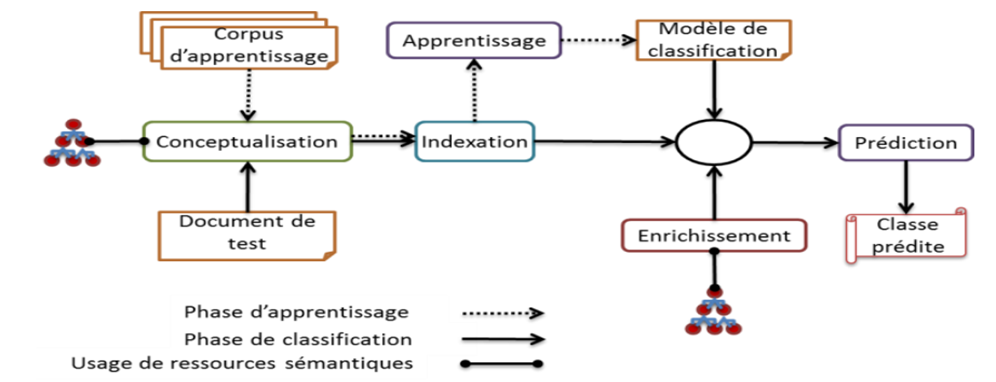
La dernière étape est la classification du texte. Cette partie a pour but de sélectionner l’informations pertinentes et rejeter les informations non pertinentes du texte reconnu par système de RAP. À la fin de cette étape, on disposera d’une base de données qui contient uniquement les anomalies rencontrées, afin de les analyser facilement et les faire remonter sur le système d’information. Quelques algorithmes de classifications sont décrits brièvement ci-dessous.

Figure 13 : Méthode de classification de texte

#### Classificateur Naïve Bayes

Naïve bayes est un classificateur génératif qui est largement utilisé dans la récupération d’information. C’est un modèle probabiliste d’apprentissage automatique qui est utilisé pour la tâche de classification. Il est basé sur l’idée que les variables prédictibles d’un modèle d’apprentissage automatique sont indépendantes les unes des autres.

La probabilité qu’un document représenté par un vecteur = (wi2, wi3, wi4, ....., appartient à une classe donné . Il calcule la probabilité par application du théorème de Bayes énoncé par:

Ainsi, pour déterminer , le classificateur Bayésien suppose que toutes les caractéristiques sont indépendante les unes des autres, la formule de calcul de la probabilité d’adhésion à un document à une classe est la suivante:

En résumé, la classification d’un document Di par ce type de classificateur consiste à calculer

Probabilité pour k = {1, 2, ....., | C |} où C est l’ensemble des classes, puis assigner le document à la classe qui génère la probabilité maximale. Ces classificateurs ont montré

Leur efficacité dans de nombreux travaux de classification de texte, en particulier le filtrage des courriels.

#### Support Vecteur machine (SVM)

L’idée générale de SVM est de représenter l’espace des exemples (textes) dans un vecteur l’espace où chaque document est un point dans cet espace, et de trouver la meilleure séparation possible de cet espace en deux classes. L’espace de séparation est une surface de décision appelée marge, définie par les points du vecteur porteur. La marge est définie comme la plus petite distance entre les exemples de chaque classe et la surface séparatrice S:

Ainsi, la décision s’appuie sur le SVM pour couper l’espace en deux: d’un côté, ce qui est dans la catégorie, de l’autre côté, ce qui n’est pas là. L’approche SVM permet donc de définir en apprenant un hyperplan dans un espace vectoriel qui sépare le mieux les données de la formation en deux classes, en minimisant le risque d’erreur et en maximisant la marge entre deux classes. La qualité de l’hyperplan est déterminée par son écart avec les hyperplans parallèles les plus proches du point de chaque classe. Le meilleur hyperplan est celui qui a la plus grande marge.

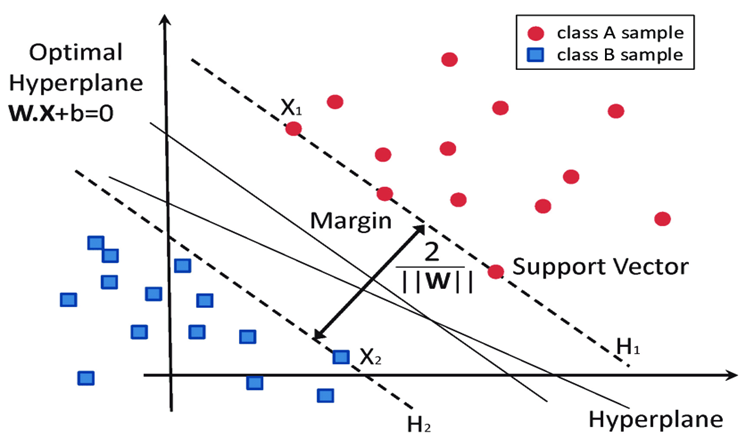


Figure 14 : Vecteur à support machine

#### Les K plus proche voisin

Le KNN est une méthode non paramétrique dans laquelle le modèle mémorise les observations de l’ensemble d’apprentissage pour la classification des données de l’ensemble de test. La phase d’apprentissage consiste à stocker les exemples étiquetés. Le classement de nouveaux textes s’opère en calculant la similarité entre les représentations vectorielles du document et celles de chaque corpus d’apprentissage. Les k éléments les plus proches sont sélectionnés, et le document est assigné à la classe majoritaire.

Le choix de la valeur de k est dépendant de la taille de l’échantillon et des classes, et influence les résultats de la classification.

#### La forêt aléatoire (Random Forest classifier)

La forêt aléatoire est un algorithme d’apprentissage supervisé qui est utilisé à la fois pour la classification et la régression. Cependant, il est principalement utilisé pour les problèmes de classification. Une forêt est composée d’arbres et plus d’arbres signifie une forêt plus robuste. De même, l’algorithme de forêt aléatoire crée des arbres de décision sur des échantillons de données, puis obtient la prédiction de chacun d’eux et sélectionne enfin la meilleure solution au moyen du vote. C’est une méthode d’ensemble qui est meilleure qu’un seul arbre de décision car elle réduit le surajustement en faisant la moyenne du résultat.

Figure 15 : Architecture d’une foret aléatoire

Le fonctionnement de l’algorithme random forest:

**Étape 1** − Tout d’abord, on commence par la sélection d’échantillons aléatoires à partir d’un ensemble de données donné.

**Étape 2** − Ensuite, cet algorithme construit un arbre de décision pour chaque échantillon. Ensuite, il obtiendra le résultat de prédiction de chaque arbre de décision.

**Étape 3** − Dans cette étape, le vote sera effectué pour chaque résultat prévu.

**Étape 4** − Enfin, on sélectionne le résultat de prédiction le plus voté comme résultat de prédiction final.

### Evaluation des performances (score)

Les scores sont aussi importants que les autres phases du processus de construction d'un classificateur. Ils permettent d’estimer la fiabilité d’un. Plusieurs mesures ou scores existent pour estimer et vérifier le degré de généralisation d'un classificateur. Toutefois, ces mesures sont à l'origine utilisées dans la recherche documentaire, puis les spécialistes du data mining les ont adaptées aux autres branches de cette science. Il existe plusieurs indicateurs qui reflètent la réussite ou l'échec d'un classificateur.

#### La matrice de confusion

La matrice de confusion en text mining sert à vérifier le bon classement des documents préalablement étiquetés. Elle indique si un classificateur fonctionne bien et à quel degré de fiabilité. Chaque classe du classificateur est représentée par une colonne et une ligne. La ligne indique le nombre de documents réels appartenant à la classe (C) et la colonne indique à quel nombre de documents cette classe (C) est assignée.

1. Vrai positif (TP)

Les valeurs réelles et prédites sont identiques. La valeur prédite du modèle est positive, ainsi qu’une valeur positive réelle.

1. Vrai négatif (TN)

Les valeurs réelles et prédites sont identiques. La valeur prédite du modèle est négative, avec une valeur négative réelle.

1. Faux positif (erreur de type 1) (FP)

Les valeurs réelles et prédites ne sont pas les mêmes. La valeur prédite du modèle est positive et faussement prédite. Cependant, la valeur réelle est négative. On peut parler d’une erreur de type 1.

1. Faux négatif (erreur de type 2) (FN)

Les valeurs réelles et prédites ne sont pas les mêmes. La valeur prédite du modèle est négative et faussement prédite. Cependant, la valeur réelle est positive. On peut désigner cette erreur comme une erreur de type 2.

1. La précision:

C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de document auxquels la classe (C) est assignée. La fonction mathématique est la suivante:

#### Le rappel

C'est le rapport entre le nombre de documents correctement classés dans la classe (C) sur le nombre de documents appartenant à la classe (C)

#### La F-mesure

La F-Mesure est un indicateur qui combine le rappel et la précision elle est donnée par la formule suivante:

## Le Chatbot (assistant virtuel)

On mettra en œuvre un Chatbot, dont le principe dans ce projet est d’interagir avec les rouleurs afin de leurs apporter de l’aide via des messages textes ou vocaux et se comporter comme s'il était capable de comprendre la conversation et répondre aux rouleurs de manière appropriée.

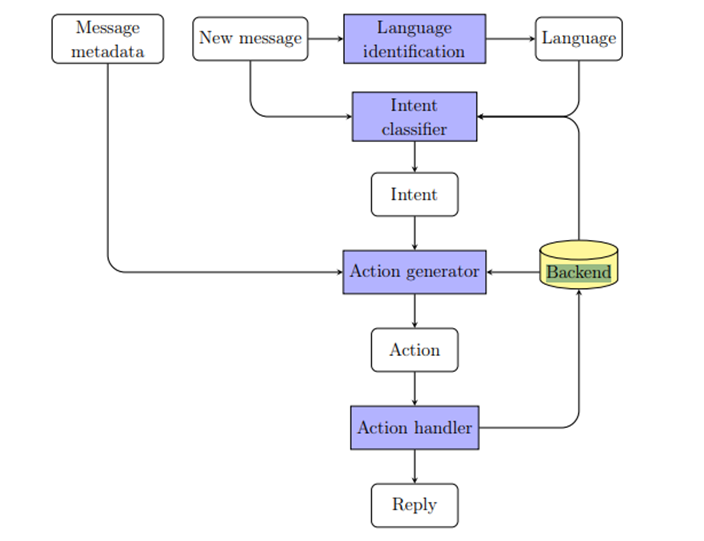
L’étape initiale de cette partie est le prétraitement des données en les nettoyant comme décrit précédemment.

Figure 16 : Architecture global d'un Chatbot

### Le Word embedding

Cette étape sert à convertir les mots du message reconnu en un vecteur de nombres réels. On va reprendre le même chemin en convertissant le message reconnu en un vecteur de nombres réels. Plusieurs algorithmes ont été développés comme expliqué précédemment.

### Identification de l’intention

Le premier problème à résoudre est le problème de classification de l’intention. Il sera modélisé comme un problème de multi classification dont la sortie est simplement l’étiquette du problème détecté. Toutefois, afin de permettre un réglage fin du comportement du bot, le classifier devrait également générer le vecteur correspondant aux probabilités de classe lors de la prédiction

Lors de la réception d’un message, l’assistant virtuel doit être en mesure d’identifier l’intention du rouleur et l’objectif qu’il tente d’atteindre. Ceci est modélisé sous forme d’un multi classificateur dont les étiquettes sont les noms des intentions possibles du rouleur, afin de déterminer la demande de l'utilisateur en fonction de plusieurs messages. On peut utiliser plusieurs approches. Quelques-unes sont décrites brièvement ci-dessous.

#### Approche bayésienne

L’approche bayésienne consiste à utiliser une mise à jour pour déduire l’intention correcte de l’utilisateur. Il est basé sur la règle de Bayes:

Dans ce contexte, c est une étiquette d’un problème du rouleur et m est le message reçu. Puisque le résultat de la mise à jour doit être un vecteur de toutes les probabilités de classe, le terme P (m) peut simplement être ignoré car il ne dépend que de m et ne sert que d’opération de normalisation. P (c) est la probabilité a priori et il est extrait directement de la sortie du classificateur d'intention. Enfin, le terme P (m | c) est estimé en utilisant le classificateur d'intention sur la réponse du rouleur.

#### Approche standard (non-bayésienne)

#### 

Cette approche se base sur des réseaux de neurones. Il existe plusieurs types d’architecture de réseau de neurones. On cite deux:

**Réseau de neurones artificiels**, qui est une application, non linéaire par rapport à ses paramètres W qui associe à une entrée x une *sortie = f(x, W).* Les réseaux de neurones peuvent être utilisés pour la régression ou la classification. Les paramètres W sont estimés à partir d’un échantillon d’apprentissage. La fonction à minimiser n’est pas convexe, ce qui conduit à avoir des minimas locaux.

Un neurone artificiel est une fonction de l’entrée x = ( pondérée par un vecteur de poids de connexion complétée par un biais et associée à une fonction d’activation φ , à savoir:

* La fonction d’activation sigmoïde:
* La fonction d’activation ReLu:
* La fonction d’activation identité:

**Un perceptron multicouche** est une structure composée de plusieurs couches cachées de neurones où la sortie d’un neurone d’une couche devient l’entrée du neurone de la couche suivante. De plus, la sortie d’un neurone peut aussi être l’entrée d’un neurone de la même couche ou d’un neurone des couches précédentes (c’est le cas pour les réseaux de neurones récurrents). Sur la dernière couche, appelée couche de sortie, on peut appliquer une fonction d’activation différente de celle des couches cachées selon le type de problèmes qu’on a à résoudre: régression ou classification.

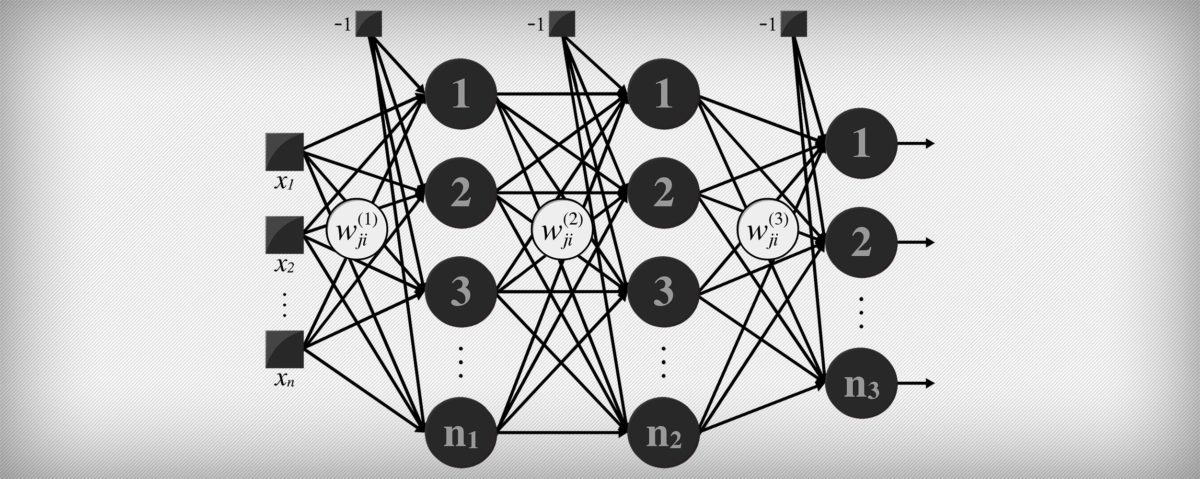


Figure 17 : Architecture d’un multi perceptron

### Génération de réponses

Afin de communiquer, un agent conversationnel doit posséder la capacité de répondre. De plus, les réponses doivent être cohérentes en fonction du contexte de la conversation. Ce problème peut être résolu en utilisant deux modules différents fonctionnant en paire: un qui génère une liste de réponses candidates et l'autre qui sélectionne la réponse la plus appropriée. Deux approches populaires ont été développés, méthodes basées sur la récupération et sur la génération.

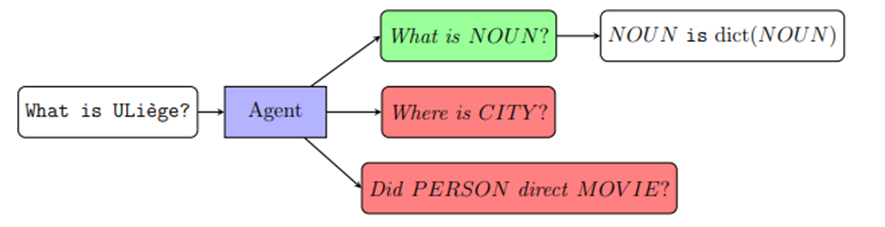
Les techniques basées sur la récupération reposent simplement sur une vaste base de données de réponses des candidats et associe ces réponses aux informations du message de l'utilisateur pour trouver la réponse appropriée. Cette information peut simplement être une expression régulière qui ressemble à des structures de phrases particulières ou peut être la sortie d'un modèle d'apprentissage automatique.

Figure 18 : Un exemple basé sur la récupération de réponse

Les techniques génératives, reposent sur des modèles génératifs pour produire de nouvelles réponses sans avoir besoin d'une vaste base de données d'exemples. Les réponses peuvent être générées facilement à condition que le modèle soit correctement formé. Toutefois à ce jour, les performances des approches génératives sont insuffisantes pour donner de bons résultats.

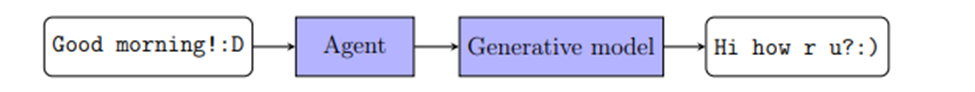


Figure 19 : exemple de génération de réponse basé sur les techniques génératives

### Evaluation des performances

Afin d’évaluer les performances d’un Chatbot, il existe plusieurs indicateurs qui reflètent la réussite ou l'échec du Chatbot. Ces métriques identifiées sont un ensemble d’outils complets qui apporte de la valeur aux utilisateurs et aide à suivre les performances globales d’un Chatbot. Ces métriques sont expliquées dans la section **[3.3.4].**

## Benchmark des solutions existantes

### Technique de word embedding

|  |  |  |
| --- | --- | --- |
| Modèle | Avantages | Risques |
| Word2Vec | -il capture la position d’un mot dans le texte (syntaxique).  -il capture le sens des mots (sémantique) | - Il ne peut pas capturer des mots hors du vocabulaire du corpus  - Il ne peut pas capturer le sens du mot du texte |
| TF-IDF | - Facile à calculer  - Mesure de base pour extraire les termes les plus descriptifs dans un document | - Il ne saisit pas la position dans le texte (syntaxique)  - Il ne saisit pas le sens dans le texte (sémantique) |
| GloVe | - Il capture la position des mots dans le texte (syntaxique)  - Il capture le sens dans les mots (sémantique) | - Consommation de mémoire pour le stockage  - Il ne peut pas capturer des mots hors du vocabulaire du corpus  - Il ne peut pas capturer le sens du mot du texte |
| FastText | - Fonctionne pour des mots rares  -Résout à partir de mots de vocabulaire avec n-gram dans le niveau de caractère | - Informatiquement est plus cher en comparant avec GloVe et Word2Vec  - Consommation de mémoire pour le stockage  - Il ne peut pas capturer le sens du mot du texte |

### Technique de classification de texte

|  |  |  |
| --- | --- | --- |
| Techniques | Avantages | Risques |
| Naïve Bayésienne | -Il requiert peu de donnée pour estimer les paramètres nécessaires à la classification  -Modèle probabiliste | -Il fonctionne mieux avec des valeurs catégoriques qu'avec des valeurs numériques  -Problème de la fréquence zéro . |
| SVM | -Performant, non biaisé par les valeurs aberrantes.  -Insensible au sur ajustement | -Ne convient pas aux problèmes non linéaires  -Pas le meilleur choix pour un grand nombre de fonctionnalités |
| KNN | -Simple à comprendre  -Rapide et efficace. | -Besoin de choisir manuellement le nombre de voisins «k» |

### Identification de l’intention (Extraction de l’information)

|  |  |  |
| --- | --- | --- |
| Techniques | Avantages | Risques |
| Naïve bayésienne | - fonctionne sur des problèmes non linéaires  -Non biaisé par les valeurs aberrantes | - Basé sur l’hypothèse que les caractéristiques ont la même pertinence statistique. |
| Standard | -basé sur les réseaux de neurone  - Rapide et efficace | - Besoin de choisir manuellement le nombre de voisins |

# Incertitudes et difficultés techniques

La RAP est une problématique complexe ou les méthodes utilisées sont très variées et fortement dépendantes du système désiré. L’environnement du projet fait ressortir plusieurs problématiques.

# Sensibilité à l’accent non-natif

La RAP non native est la RAP dans un contexte où un locuteur parle une langue différente de sa langue maternelle/natale. C’est un domaine à part de la RAP, toujours sujet à des travaux de recherche, dont les performances sont beaucoup plus faibles que la RAP native. Le monde de l’industrie et du bâtiment comportant un nombre important de locuteurs non-natifs, il faut que le système de reconnaissance de la parole (SRAP) soit en capacité de les comprendre.

* 1. **Le système de reconnaissance en offline**

Sur les chantiers une connexion internet n’est pas toujours disponible. Le SRAP ne peut donc pas s’appuyer sur un principe d’enregistrement – requête au serveur – reconnaissance de texte au niveau du serveur – retour du texte au système comme Siri d’Apple ou Alexa d’Amazon. Il faut donc que l’outil possède toutes les composantes nécessaires à la RAP localement. Cela a pour conséquence l’obligation d’avoir des modèles assez légers pour être supportés par la tablette.

* 1. **Passer de l’audio à l’extraction d’information**

En passant du texte à l’extraction, le texte reconnu de chaque audio sera enregistré. Cette étude s’avère volumineuse en termes de stockage. Donc, il est nécessaire que le système puisse avoir les fonctionnalités nécessaires pour pallier ce problème.

* 1. **La rapidité du système RAP**

Une des principales caractéristiques du système est sa forte interactivité avec l’utilisateur. Le SRAP doit donc être capable de comprendre les commandes utilisateurs en un temps très court ne dépassent pas quelques millisecondes. Ceci entraîne une difficulté supplémentaire car cela induit que l’ensemble des calculs effectués par le SRAP soit terminé dans la seconde. Des choix de méthodes de calculs doivent donc être faits afin d’optimiser le temps sans dégrader la précision.

* 1. **L’ensemble des données**

Avoir un bon modèle nécessite au minimum 10 000 données en termes de quantité et la qualité de celles-ci impacte directement la qualité du modèle. De plus, actuellement il n’existe pas de norme universelle quant au format des données utilisées dans le cadre de la RAP. En fonction des moyens utilisés lors de l’enregistrement, de la façon de stocker les données ou encore de choix propre au créateur du jeu de données, les formats de ces ensembles de données peuvent être très différents.

* 1. **La résistance aux bruits**

Dans les pistes de roulage, le bruit influence la précision du modèle. Donc faut que l’outil soit résistant aux conditions d’environnement variables.

# Présentation des travaux de R&D

## Démarche de travail

La première étape a été de définir les limites du projet et l’orientation à donner pour la recherche.

* L’interface vocale doit être capable de comprendre l’utilisateur, de réagir à ses commandes et d’avoir le rôle de support privilégié par l’utilisateur dans son interaction avec l’outil. Cette interface doit faire l’objet d’un travail de recherche.
* On souhaite pouvoir rediriger progressivement la réponse du rouleur vers des caractéristiques prédéfinies en utilisant la méthode de « text mining » ou « extraction de connaissance ».
* L’incertitude porte sur la capacité du modèle à comprendre toute la divergence de réponses possibles qui n’ont pas pu être prévues au préalable. Le but est de pouvoir poser la question suivante en adéquation avec la réponse donnée par le rouleur, et de renseigner les informations sur le défaut en mettant à jour la base de données logicielle. Donc faut que le système possède un Chatbot qui prendra la place d’un assistant.

La partie recherche se décompose en plusieurs étapes:

-Etat de l’art sur la reconnaissance de la parole, sur l’extraction d’informations pertinentes avec le texte mining et sur le Chatbot (assistant virtuel)

-Faire un benchmark de toutes les solutions existantes et des outils d’open source qui pourront être utiliser toute en respectant les limites du projet.

-Collecte de données nécessaires au projet.

-Expérimentation et comparaisons des résultats.

-Les perspectives et prévoir la continuité du projet ainsi que les débouchés et les opportunités qu’il a générés.

## Discussion de l’ancienne version

Les techniques de reconnaissance vocale qui ont été développées dans les anciennes recherche (V1) fonctionnent avec une précision de 40%. La principale raison de cette dégradation de précision est que le système RAP est confrontés à des locuteurs non-natifs qui s’expriment dans des langues maternelles différentes.

Toutefois, Le but initial de ce projet est de proposer de nouvelles approches pour améliorer la robustesse de l’interface homme/machine face à la parole non-native, en apportant des corrections dans la phase de traitement des signaux audio. Il s'agit de modifier un système de RAP d'une langue cible afin de le rendre plus sensible aux erreurs que produisent des locuteurs non-natifs.

## Description des travaux réalisés pour la reconnaissance vocale

### **Benchmark des solutions existantes de reconnaissance vocale**

Pour la partie du DeepSpeech, il a été décidé d’effectuer une recherche des outils Open Source disponibles sur le marché. Ci-dessous la liste non exhaustive des outils avec leurs caractéristiques principales:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solution | Connection | Langage | Gratuit test | Gratuit Commerce | Langue |
| Google |  |  |  |  |  |
| Alexa |  |  |  |  |  |
| Apple |  |  |  |  |  |
| CMU PocketSphinx |  | Java,C,Python,others |  |  | *English plus 10 autres langues* |
| Kaldi |  | C++,Python |  |  | *English* |
| Julius |  | C ,Python |  |  | *Japanese* |
| HTK |  | C ,Python |  |  | *Pas de langue prédéfinie* |
| Snips |  | Python |  |  | *English*  *French*  *German Japanese Italian*  *Spanish* |
| Mozilla DeepSpeech |  | Python |  |  | *English* |
| Wav2letter++ |  | C++ |  |  | *Pas de langue prédéfinie* |
|  |  |  |  |  |  |
| DeepSpeech 2 on PaddlePaddle |  | C++ |  |  | *English*  *Mandarin*  *+autres langues* |

Tableau 3: Benchmark outils Open Source

En prenant en compte les critères définis en amont et l’environnement, il a été décidé d’utiliser l’outil de Mozilla, Mozilla DeepSpeech.

### Mozilla DeepSpeech

Mozilla DeepSpeech est un système de RAP « end-to-end », utilisant un modèle formé par des techniques d'apprentissage automatique basé sur le document de recherche Deep Speech de Baidu. Ce système est connu par sa robustesse aux bruits et aussi par le fait qu’il n’est pas nécessaire de définir de lexique de prononciation.

Mozilla DeepSpeech a l’avantage d’être codé en langage Python, permettant ainsi des modifications si nécessaires, de pouvoir entraîner son propre modèle avec une certaine souplesse dans le paramétrage de l’entraînement et de ne pas nécessiter de connexion internet pour fonctionner car le modèle est utilisé localement.

Il prend en entrée le modèle de langage, l’alphabet, ainsi que les données nécessaires à la création du modèle et ressort en sortie le modèle final. Ce modèle, après validation, est utilisé par l’outil pour reconnaître le texte du signal à traiter.

La qualité du modèle est décidée selon trois critères:

-Le taux d’erreur au mot (Word Error Rate noté WER).

-Le taux d’erreur au caractère (Character Error Rate noté CER).

-La perte (loss).

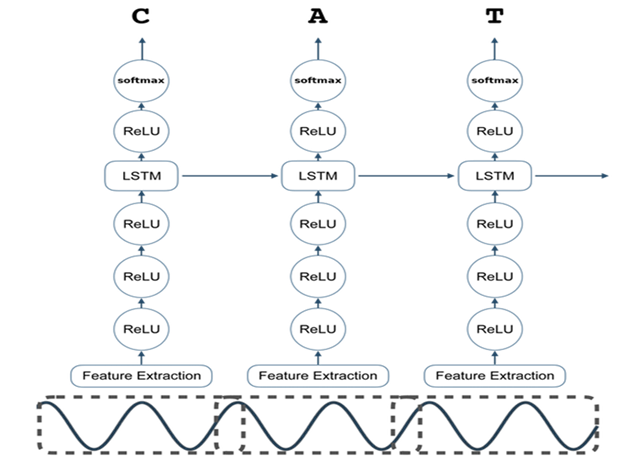
Le cœur du moteur est un réseau neuronal récurrent (RNN) formé pour intégrer des spectrogrammes vocaux et générer des transcriptions de texte

Figure 20 : Architecture RNN

Soit un seul énoncé x et étiquette y être échantillonné à partir d’un ensemble de formation.

S={(),(),...}.

Chaque énoncé,  est une série chronologique de longueur où chaque tranche de temps est un vecteur de fonctionnalités audio,  où *t*=1,…, .On utilise les MFCC comme caractéristiques ; donc dénote la *p-ème* fonction MFCC dans la trame audio à la fois *t*. Le but des RNN est de convertir une séquence d'entrée *x* en une séquence de probabilités de caractère pour la transcription *y*, avec  =P(∣x) , où pour le français ∈{a,b,c,...,z,espace,apostrophe,vide}. (L'importance du vide sera expliqué ci-dessous.).

Le modèle RNN est composé de 5 couches d'unités cachées. Pour une entrée *x*, les unités cachées au niveau du calque  « *l »* sont notés  avec la convention est l'entrée. Les trois premières couches ne sont pas récurrentes. Pour la première couche, à chaque foist *t*, la sortie dépend de la trame MFCC  avec un contexte de*C* cadres de chaque côté (C=9).Les couches non récurrentes restantes opèrent sur des données indépendantes pour chaque pas de temps. Ainsi, pour chaque fois *t,* les 3 premières couches sont calculées par:

= g ( ++)

Notez que doit être calculé séquentiellement à partir de t=1 à t=pour le i-th énonciation.

La cinquième couche (non récurrente) prend les unités avancées comme entrées:

= g ( + )

La couche de sortie est constituée de logits standard qui correspondent aux probabilités de caractères prédites pour chaque tranche de temps t et caractère k dans l’alphabet:

= +

Ici dénote la k-th biais et  ,le k-th élément du produit matriciel.

Une fois qu’on a calculé une prédiction pour , on calcule la perte de CTC *L*(,y) pour mesurer l'erreur de prédiction. (La perte de CTC nécessite le vide ci-dessus pour indiquer les transitions entre les personnages.) Pendant l'entraînement, on peut évaluer le gradient ∇ *L*(,y) par rapport aux sorties du réseau étant donné la séquence de caractères de la vérité au sol *y*. À partir de ce point, le calcul du gradient par rapport à tous les paramètres du modèle peut se faire par rétro propagation à travers le reste du réseau. La méthode Adam est utilisée pour l’entraînement.

### Langues et dataset

Le choix de la langue est impacté par deux critères: la disparité de volume de données disponibles entre chaque langue et la taille du vocabulaire que l’on veut reconnaitre.

#### Le dataset Common Voice de Mozilla

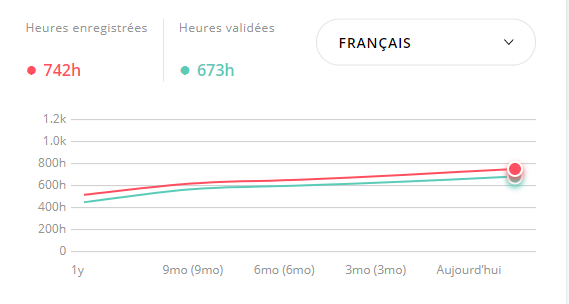
Le corpus Common Voice est une collection massivement multilingue de discours transcrit destiné à la recherche et au développement des technologies de la parole. Le corpus actuel contient 623 h validées.

Figure 21 : Répartition des langues du Dataset

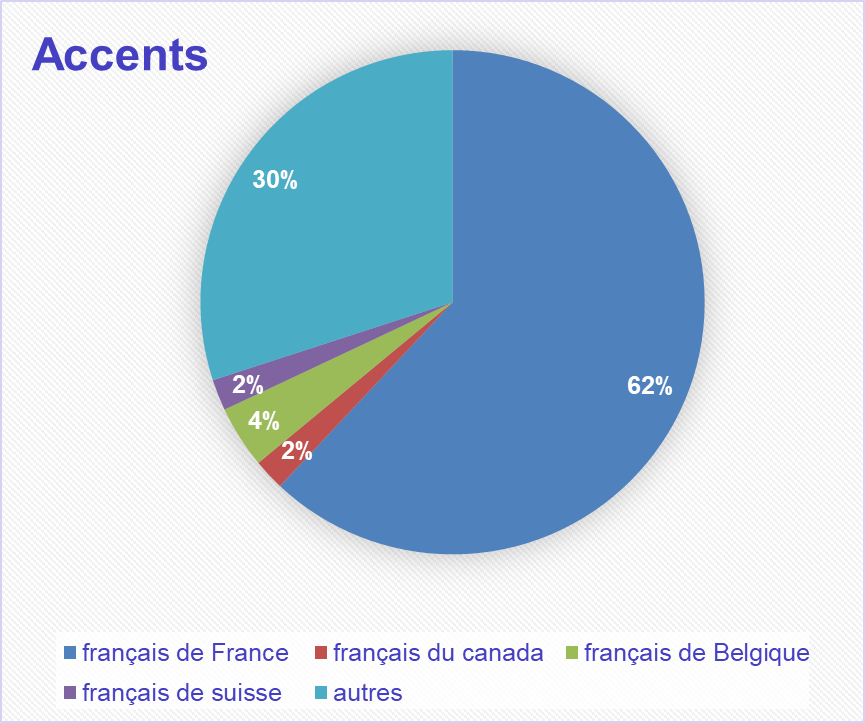
Pour la communauté francophone, il a été détecté une trentaine d’accents. Il faut prendre conscience que tous les types de voix sont importants. Et, concernant le français, c’est une langue riche, avec des parlers différents selon les régions, le continent ou le pays où on le pratique. Le dataset contient plusieurs accents du français présenté dans le graphe ci-dessous.

Figure 22 : les accents en français

Le dataset contient six fichiers TSV qui représente un segment différent des données vocales, les six ayant les en-têtes de colonne suivants: [id client, chemin, phrase, votes\_up, votes\_down, âge, sexe, accent]

Les trois premières colonnes font référence à un identifiant anonyme pour le locuteur, à l'emplacement du fichier audio et au texte lu. Les deux colonnes suivantes contiennent des informations sur la façon dont les auditeurs ont<audio, transcription> paire. Les trois dernières colonnes représentent des données démographiques qui ont été éventuellement auto-déclarées par le locuteur de l’audio. Les répartitions par langue ont été effectuées le plus près possible de 70% de train, 20% de développement (ou validation) et 10% de test.

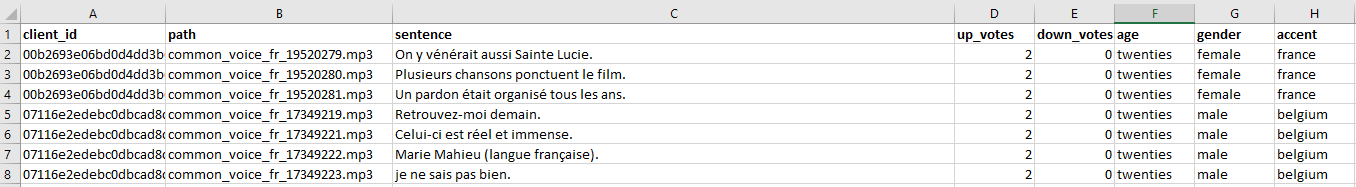


Tableau 2 : Exemple donnée fichier TSV

#### Choix de la taille du vocabulaire du corpus

La taille du vocabulaire est nécessaire pour remplir les objectifs de ce projet. La première approche a été de créer un modèle au vocabulaire très large, couvrant une grande partie du langage cible afin d’obtenir un système capable de comprendre une dictée et un large panel de commandes.

La seconde approche a été d’utiliser un corpus avec un vocabulaire très restreint, comportant uniquement les commentaires personnalisés au roulage. Cette méthode a l’avantage de ne pas demander de grosses capacités ni d’un temps important de calcul. Elle permet aussi d’obtenir rapidement des résultats satisfaisants ainsi qu’un modèle simple et léger. L’inconvénient étant qu’il faut créer le dataset à utiliser pour créer le modèle, car il n’existe pas de dataset le permettant en Open Source.

Ces deux approches possédant chacune leurs forces et faiblesses, aucune des deux n’a été mise de côté. Le choix a été fait de les développer toutes les deux en parallèle.

### Expérimentation

#### Reconnaissance de parole sur un corpus de large vocabulaire LVCSR

Il a été décidé d’utiliser l’apprentissage par transfert(TL). TL est la capacité d’un système à reconnaître et appliquer des connaissances et des compétences apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes.

L’idée derrière l'apprentissage par transfert de reconnaissance vocal permet de supprimer certaines couches d'un modèle préformé, d'initialiser de nouvelles couches pour les données de français, d'assembler les anciennes et les nouvelles couches et de mettre à jour toutes les couches via la descente de gradient. Il suffit de supprimer la couche de sortie préformée) et réinitialiser les paramètres pour correspondre au nouvel alphabet celui du français. L’objectif est que le modèle soit plus sensible aux différents accents.

Le jeu de données utilisé est un jeu de données issu de Common Voice. Ce jeu de données est composé de 623h d’enregistrements. L’accent des locuteurs est varié car ceux-ci sont d’origines diverse. L’avantage de ce jeu de données est qu’il est conçu pour fonctionner avec Mozilla DeepSpeech, permettant ainsi d’éviter une étape de préparation de données supplémentaire.

#### Reconnaissance de la parole sur un corpus de faible vocabulaire (SVCSR)

#### 

Une autre approche est de créer un corpus vocabulaire restreint qui permet de répondre aux exigences du projet. Ce corpus sera plus léger pour l’entraînement et peut gagner en précision du modèle de reconnaissance.

Ce corpus contiendra une liste de mots et de phrase courtes personnalisées au vocabulaire du roulage, ce qui va permettre de garder un vocabulaire court pour ainsi réduire le temps de création de ce corpus.

Un autre avantage d’utiliser une liste de ce petit corpus est l’utilisation de 1-gram au lieu d’un 2-gram/3-gram/4-gram. Plus n est grand, plus la taille du LM généré est grande et donc plus le modèle global du système de RAP est lourd. Le choix du type de n-gram a donc un impact direct sur le poids de l’application. L’usage d’un 1-gram permet donc de garder un modèle léger et correspond le mieux à cette approche. L’autre bénéfice est d’avoir une certaine rigidité sur les mots reconnus, la liste étant courte et les mots phonétiquement différents, réduisant ainsi le WER.

Ces n-gram sont générés en utilisant la bibliothèque KenLM (L’utilisation d’une variante de cette bibliothèque a été nécessaire pour créer le 1-gram).

#### Préparation des données

Le jeu de données contient:

* Les six fichiers \*.tsv produits pour la langue téléchargée.
* Les fichiers audio mp3.

Pour préparer ces données sous une forme que DeepSpeech comprend, on exécute la commande (bin/import\_cv2.py): cette commande sert à ajouter des \*.wav aux mp3 plus des fichier \*.csv. La commande décrite ci-dessus permet de découper les données en un ensemble d’entraînement, de dev et de test:

* Train.csv (70%)
* Dev.csv (20%)
* Test.csv (10%)
* Validated.csv
* Train-all.csv
* Other.csv

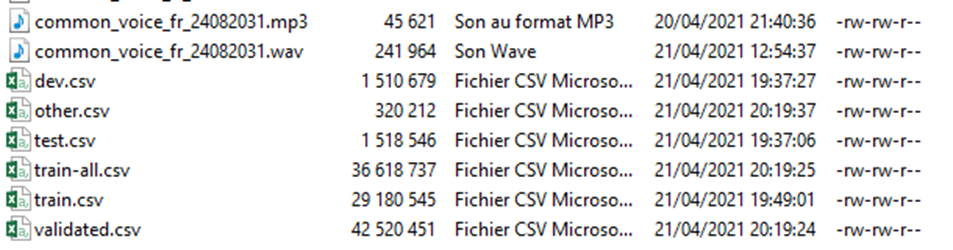
******

Figure 23 : fichiers CSV

Toutes les fichiers \*.CSV font référence à leurs échantillons par des chemins absolus. Ainsi, le déplacement de ce sous-répertoire nécessiterait une autre importation ou une modification des fichiers CSV en conséquence. Pour utiliser exécuter DeepSpeech.py pour la formation, on lui donne un chemin vers les fichiers \*.csv comme ceci:

--train\_files data/frn/clips/train.csv;

--dev\_files data/frn/clips/dev.csv ;

--test\_files data/frn/clips/test.csv ;

#### Création d’un modèle de langage

Le choix du type de n-gram a un impact direct sur le poids de l’application. Les n-gram sont générés en utilisant la bibliothèque KenLM. Pour générer le kenLM on se base sur le texte de Wikipédia « vocabulaire.txt » (500 000 mots) utilisée par Mozilla DeepSpeech et par la suite on crée KenLM. L’utilisation d’une variante de cette bibliothèque a été nécessaire pour créer le 1-gram car l’originale ne le prend pas en compte.

Pour générer « lm. binary » et « vocab-500000.txt» Ensuite, on utilise le generate\_lm.py qui se trouve dans le dossier nommé « lm ». Puis, on utiliser generate\_package.py pour générer le package de scorer « KenLM.scorer » à l'aide des fichiers « lm.binary » et « vocab-500000.txt ».

#### Relancer l’ancienne version (version 2020)

Dans un premier temps, on va relancer l’ancienne version (précédente recherche) avec la nouvelle base de données. Ensuite on va comparer les résultats pour voir si la précision augmente uniquement en rechargeant la base de données.

#### Premier entraînement sur LVCSR :

-Carte Graphique: Quadro P2000

-Système d'exploitation: Windows

-Putty: émulateur de terminal Linux pur exécuter des scripts

-fileZila: est un client FTP, FTPS et SFTP

##### Reconnaissance de parole sur un corpus de large vocabulaire LVCSR

L'objectif est de créer un modèle français de Mozilla DeepSpeech et l’adapter à l'accent non natif en utilisant le dataset CommonVoice français. Ce dataset est séparé en trois parties train, dev et test. On souhaite avoir un modèle qui est en capacité de reconnaître les phrases ou les mots prononcés et l’optimiser en diminuant les taux d’erreurs des mots (WER) et des caractères(CER).

##### Approche TL

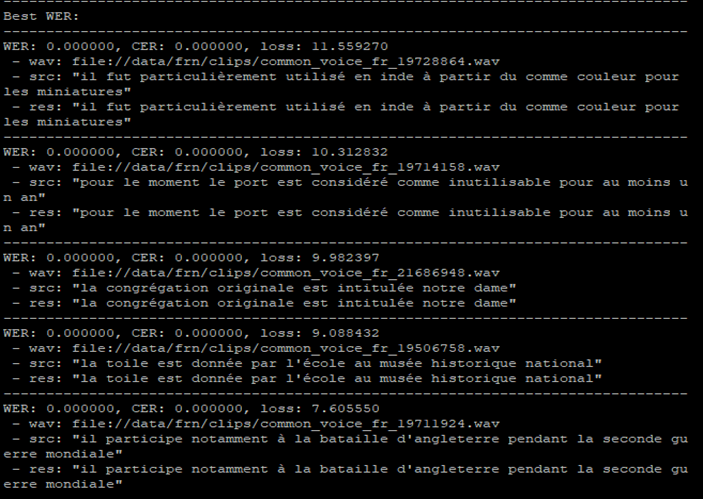
Pour faire usage du TL, on utilise une variante de Mozilla DeepSpeech appelée DeepSpeech TL. On a supprimé la dernière couche, celle de sortie, et on a réinitialisé les hyper-paramètres avec le nouvel alphabet. On a exécuté la commande TL pour entraîner le modèle.

On a entraîné le modèle avec les nouvelles données de Common Voice, et avec le modèle de langage de Kenlm crée avec le vocabulaire ancien de « 115 025 mots », on a obtenu un WER de 67,6% et un CER de 36,5%. Ce résultat n'est pas satisfaisant, mais il a augmenté légèrement par rapport à l’ancienne version. Cela montre bien la nécessité de sensibiliser le modèle à l'accent non natif et à la parole bruitée. Afin d’améliorer la précision du modèle, il est nécessaire d’optimiser les hyper-paramètres et de changer le modèle de langage toute en créant un nouveau avec le vocabulaire générer sur Wikipédia de « 500 000 mots ».

##### Résultat d’entraînement pour un « batch\_size =256 » et « epoch=2 »

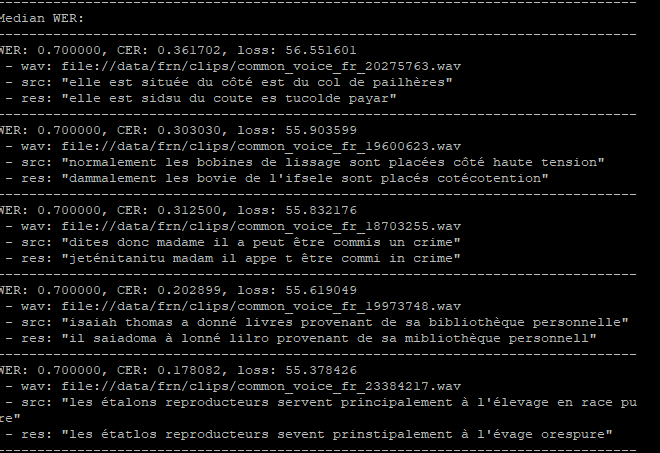
* Best WER

Dans cette partie le modèle a réussi à connaître quelques audio avec un taux d’erreur %WER et %CER de zéro.



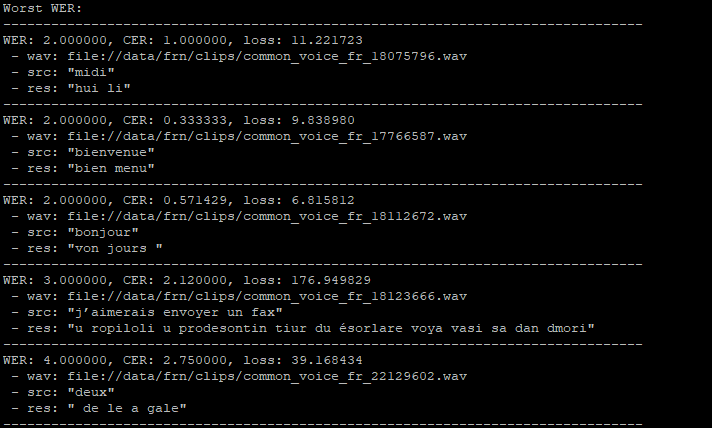
* Media WER

Dans cette partie le modèle a pu reconnaître les mots avec un taux d’erreur moyen WER 70% et CER 17%-36%



* Worst WER

Dans cette partie on va voir le mauvais résultat. Le taux d’erreur WER 200%-400%

CER 33%-200%. Ces résultats sont dû à plusieurs facteurs qu’on décrira par la suite.

##### Comparaison avec l’ancienne version(epoch=2)

|  |  |  |
| --- | --- | --- |
|  | %WER | %CER |
| Avec l’ancienne base de donnée | 69 | 40 |
| Avec la nouvelle base de donnée | 67 | 36 |

Discussion des résultats :

On constate que les taux d’erreurs baissent légèrement. Cela est dû aux modèles de langage avec le vocabulaire de 115 025mots, et aussi le modèle n’a pas pu beaucoup s’entraîner (l’entraînement est fait que sur deux epochs).

#### Entraînement du modèle avec le nouveau paramétrage :

##### Création d’un nouveau modèle de langage:

Le modèle de langue KenLM a été généré à partir du texte normalisé de formation LM LibriSpeech. Ensuite on exécute le script pour générer les dépendances nécessaires pour la formation à savoir le vocabulaire « 500 000 mots », « lm\_binary » et « generate\_score\_package ».

Pour l’entraînement du modèle, on a réglé quelques hyper-paramètres :

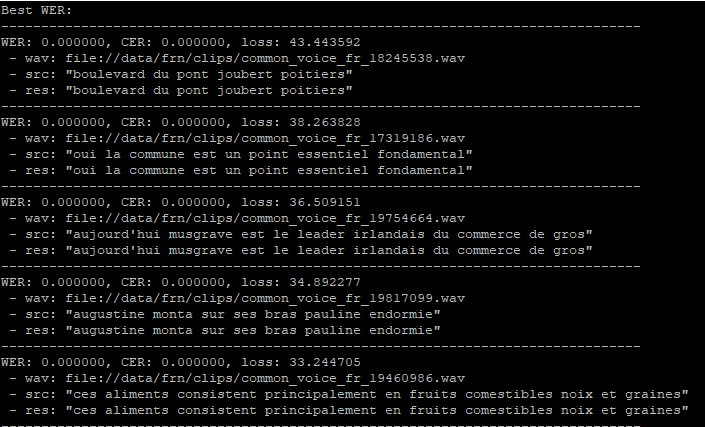
* dropout\_rate=0,4: qui sert à geler quelque neurone de la couche caché pour un entraînement rapide et efficace
* early\_stopping: sert à éviter le sur-apprentissage
* n\_hidden =2048: le nombre de neurone de la couche caché
* lm\_beta , lm\_alpha :
* Epoch: le nombre de fois que le modèle voit le jeu de données
* Batch\_size =256 fichiers: La taille des lots est l'un des hyper-paramètres les plus importants à régler dans les systèmes modernes d'apprentissage en profondeur. On souhaite utiliser une taille de lot plus grande pour entraîner le modèle, car cela permet des accélérations de calcul à partir du parallélisme des CPU. Cependant, une taille de lot trop importante entraînera une mauvaise généralisation.

##### Résultat d’entraînement :

1. Epoch=1

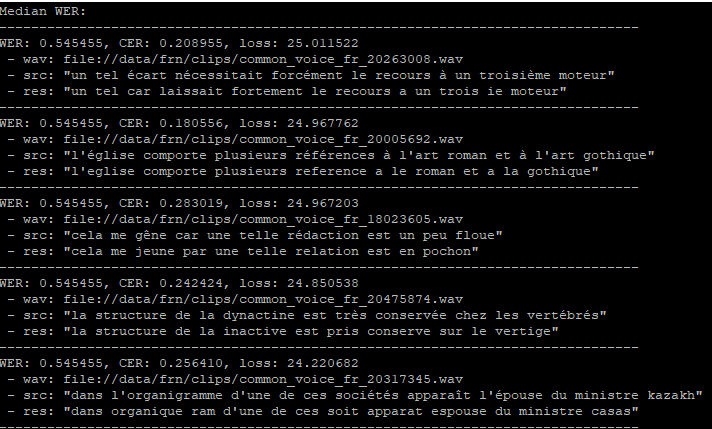
* Best WER :

Dans cette partie le modèle a réussi à connaître quelques audio avec un taux d’erreur %WER et %CER de zéro.



* Median WER :

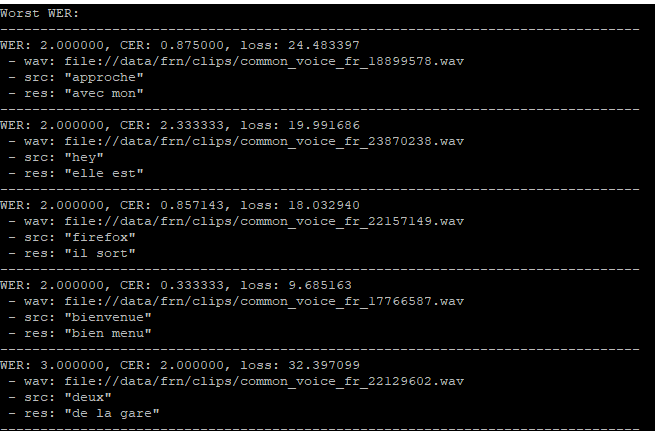
Dans cette partie le modèle a pu reconnaître les mots avec un taux d’erreur moyen WER 0,54% et CER 18%-28%



* Wrost WER :

Dans cette partie on va voir le mauvais résultat. Le taux d’erreur WER 200%

CER 33%-200%. Ces résultats sont dû à plusieurs le bruit et aux accents.



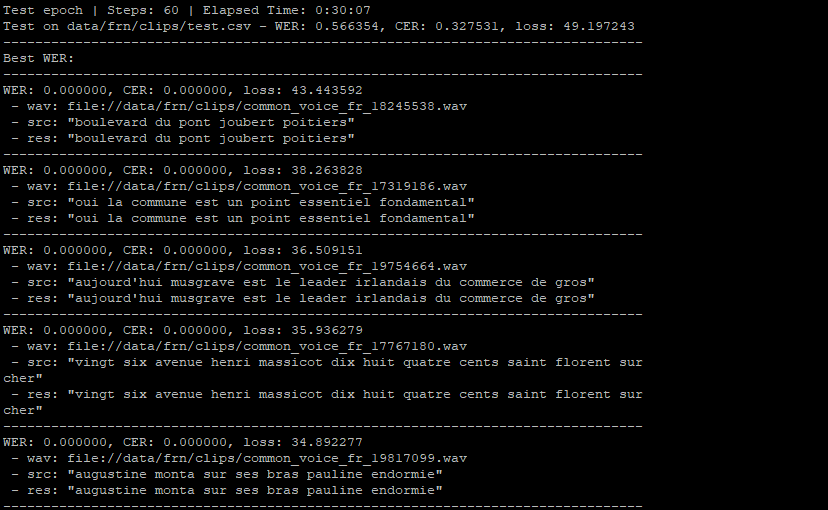
Discussion des résultats :

On constate qu’avec le réglage des hyper-paramètres, le taux d’erreurs WER a baissé de 67.8% à 55% et CER de 40% à 30% avec seulement un epoch, ce qui donne une précision du modèle de 45%. Ce résultat est satisfaisant on estime avoir un meilleur résultat avec plusieurs epochs.

1. Epoch =3

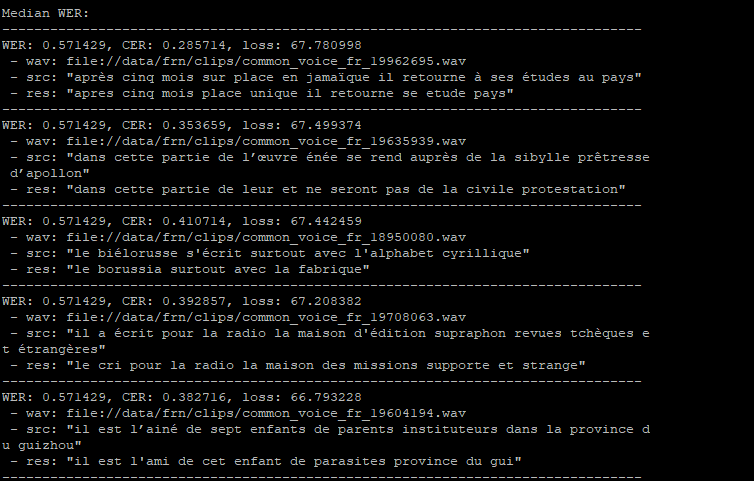
* Best WER

Dans cette partie, le modèle a réussi à reconnaître quelques audio avec un taux d’erreur %WER et %CER de zéro.



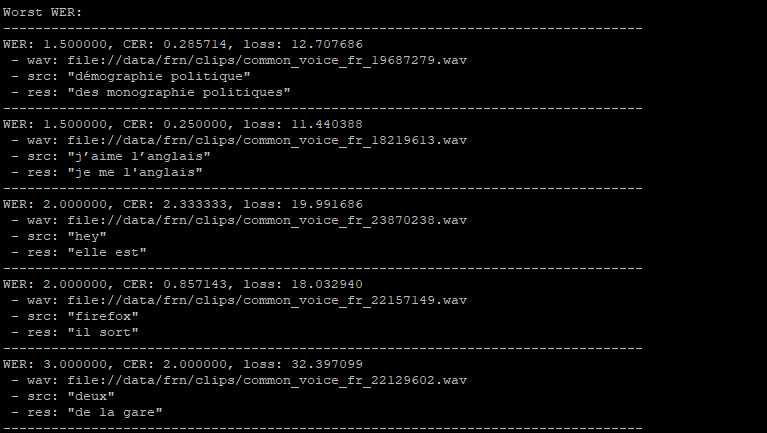
* Median WER

Dans cette partie le modèle a pu reconnaître les mots avec un taux d’erreur moyen WER 0,57% et CER 28%-41%



* Wrost WER

Le wrost présente le mauvais résultat avec un taux erreur très élevé. Le taux d’erreur WER 150%-300% CER 28%-230%.



Discussion des résultats :

On remarque que le nombre d’epochs a un impact positif sur la précision du modèle. Pour trois epochs on a obtenu le WER de 56% et le CER de 32%. Ces résultats sont dû aux bruits.

##### Impact de nombre d’epochs sur la précision du modèle

Dans cette partie, on fait varier le nombre d’epochs avec une taille de lot de 256. Le paramètre epochs permet de définir le nombre de cycles d’apprentissage (epoch en anglais). On a choisi ce batch\_size pour entraîner le modèle plus rapidement vu la contrainte de temps car cela permet des accélérations de calcul à partir CPU.

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre epoch | %WER-batch\_size:256 | %CER–batch\_size:256 | Temps d’entraînement |
| 1 | 59,7 | 41,1 | 20h |
| 2 | 57,4 | 34,6 | 1 jour et 15 h |
| 3 | 56,6 | 31,3 | 3 jours 12 h |
| 4 | 55,25 | 30,9 | 5 jours et 14h 44 min |
| 5 | 54,54 | 25,3 | 7 jours 2h 33 min |

Tableau 3 : Impact de nombre d’epoch sur la précision

Figure 24 : Evolution de WER et CER en fonction de nombre d'epoch

Discussion :

On remarque qu'à partir de quatre epochs, le modèle génère un WER variant de +/- 1% autour de 54%. Les résultats présentent une zone de variation fortement bruitée. Le WER commence à baisser dès le deuxième epoch, ce qui confirme que le modèle devient plus précis. La variation de nombre d’epochs a donc un impact positif sur la précision du modèle.

Après avoir effectué les différents tests, une observation ressort: il existe des résultats absurdes qui dégradent la qualité du modèle. Ils sont dus à un problème de qualité de l’audio ou d’incompatibilité avec le modèle de langage. Voir la figure ci-dessous

*WER: 0,571429, CER: 0,353659, loss: 67,499374*

*- wav: file: //data/frn/clips/common\_voice\_fr\_19635939*

*- src: "dans cette partie de l’œuvre* Énée *se rend auprès de la sibylle prêtresse d’Apollon"*

*- res: "dans cette partie de leurs et ne seront pas de la civile protestation"*

Figure 25: Exemple de résultat absurde

Certains résultats paraissant absurdes, ils peuvent être causés par un accent très prononcé, un enregistrement avec une discussion en arrière-plan ou à la vitesse d’élocution (débit lent ou rapide). Ces résultats sont gardés tels quels parce qu’ils sont un bon moyen de tester le modèle. Retirer ou corriger ces données de mauvaises qualités permet de diminuer le WER du modèle. L'impact de ce nettoyage n'est pas important car il ne concerne qu'un nombre peu élevé d'enregistrements.

##### Impact de la taille du lot sur la précision du modèle

Dans cette partie, on fait varier le batch\_size pour trois epochs. Le paramètre batch\_size permet de définir la taille de chaque lot (batch en anglais) pour les phases d'entraînement, validation et test. Le concept derrière est que si la taille du lot est suffisamment grande, le lot fournira une estimation stable du gradient du dataset tout en réduisant le temps de calculs. Une taille de lot trop importante entraînera une erreur OOM (Out Of Memory) si les ressources (CPU/GPU) sont insuffisantes. La taille optimale des lots pour que le modèle puisse commencer à s’entraîner avec est de 256. Au-delà de cette taille, une erreur OOM se produit.

|  |  |  |  |
| --- | --- | --- | --- |
| Batch\_size | %WER –epoch3 | %CER –epoch3 | Temps d’entrainement |
| 256 | 56,6 | 31,3 | 3 jours et 12h |
| 128 | 55,9 | 30,3 | 4 jours 20 h 39 min |
|  |  |  |  |

Tableau 4 : Impact de la taille du lot sur la précision

On remarque que la taille du lot a un impact sur la précision du modèle. Un test est donc effectué en tenant compte de la variation de la perte (loss en anglais) entre chaque étape de validation. Le principe est le suivant: après chaque étape de validation un checkpoint est enregistré. Le meilleur checkpoint est celui avec la fonction de perte(loss) minimum. Il est utilisé par la suite pour construire le modèle. On a pas pu continuer l’entraînement du modèle à cause des problemes liés aux ressources.

Pour conclure, on estime qu’une taille de lot plus petite conduit généralement non seulement à un plus grand nombre d'itérations d'un algorithme d'apprentissage, mais aussi à une précision globale plus élevée, c'est-à-dire, un réseau neuronal qui fonctionne mieux, dans un temps d'entraînement plus important.

### Conclusion global de la reconnaissance vocale :

En conclusion, on a exploré qu’une seule approche qui est LVSCR visant à traiter l'ensemble d'un langage(LVSCR). Le modèle s'est adapté au contexte d'accent non natif et bruité. On a amélioré le modèle en modifiant les hyper-paramètres (taille de lot et le nombre d’epochs) On est donc passé d'un modèle avec un WER de 67% à un WER de 54,54%.

Afin d’améliorer la précision du modèle, il est possible d’optimiser les hyper-paramètres liées au décodeur CTC (voir tableau), cela demande beaucoup de temps de calcul.

|  |  |
| --- | --- |
| Hyper-paramètres | Définition |
| Beta1 | Paramètre beta1 d’Adam |
| Beta2 | Paramètre beta2 d’Adam |
| Epsilon | Paramètre epsilon d’Adam |
| N\_hidden | Nombre de nœuds par couche du modèle |
| Lm\_alpha | Hyper-paramètre alpha du décodeur CTC – influe sur le poids du modèle de langage |
| Lm\_beta | Hyper-paramètre beta du décodeur CTC – influe sur le poids d’insertion des mots |

Tableau 5 : Descriptif des hyper-paramètres de décodeur CTC

## Description des travaux réalisés pour text mining

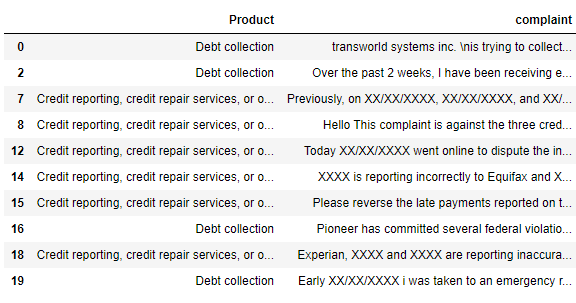
### Text mining : classification automatique de texte

Le Text Mining se définit comme l’ensemble des méthodes visant à analyser des contenus textuels bruts, tels que les textes remontés par les rouleurs, pour en tirer les informations pertinentes.

Le processus de classification d’une collection de textes consiste à étiqueter chaque texte avec une ou plusieurs classes (catégories) prédéfinies par le biais d’un algorithme d’apprentissage automatique (Machine Learning / Deep Learning).

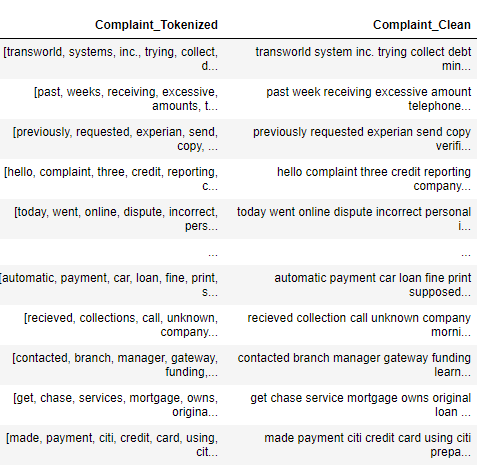
### Importer et charger le fichier de donnée

Le fichier de données se présente sous format csv. Le fichier contient deux colonnes (catégorie et la description).

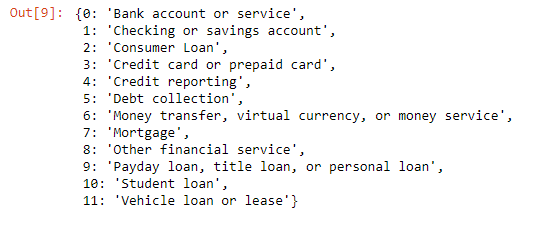


### Le prétraitement des données

Après avoir appliqué le prétraitement des données on a obtenu le résultat suivant :



Enfin, on doit transformer les catégories (la cible) en valeurs entières distinctes représentant les valeurs catégorielles initiales. Le résultat est le suivant :



Après avoir effectué les étapes de prétraitement mentionnées ci-dessus. Les deux colonnes qu’on va utiliser dans les modèles d’apprentissage automatique seront le « Complaint\_Clean », représentant la plainte après la tokenisation, le nettoyage et la lemmatisation, et le « Product\_Id » qui correspond à l’une des catégories présentées ci-dessus.

### Création des données de formation et de test

On utilise un fractionnement de 80 % de l’ensemble d’apprentissage et des 20 % restants comme ensemble de test.

Avant de passer à la face de création du modèle, on doit vérifier le nombre de catégories dans chacun des ensembles de formation/test. On s’attend à ce qu’ils aient été distribués de manière égale pour les ensembles d’entraînement et de test.



On constate qu’on a divisé les catégories(classes) uniformément entre l’entraînement et le test. 12 classes dans le test et 12 classes dans l’entraînement.

### Le word embedding

À ce stade, on a nettoyé et prétraité le jeu de données, et reformaté d’une manière qui permettra de développer les algorithmes de prédiction. La dernière partie du pipeline de prétraitement consiste à représenter les textes dans un format que les ordinateurs peuvent comprendre.

#### TF-IDF

TFIDF est une méthode couramment utilisée dans l’analyse de texte et la récupération d’informations pour comprendre la pertinence d’un mot dans un document. L’idée générale est que les mots qui apparaissent plusieurs fois dans un document obtiennent une signification plus élevée. D’autre part, les mots courants comme « et » et « le » sont généralement communs dans tous les documents et ne fournissent donc pas beaucoup d’informations.

Le calcul de TFIDF est facilement appliqué en tirant parti de la fonctionnalité de la bibliothèque « sklearn ».

### Création et évaluation de modèles

La méthode d’évaluation pour les algorithmes d’apprentissage automatique, on va utiliser la « précision » comme mesure principale. Les classificateurs utilisés sont Naïfs bayésien, Support à Victor machine, régression logistique et la foret aléatoire.

On entraîne différents modèles sur le jeu de données et on sélectionne celui qui produit les meilleures performances.

Un facteur important dans les performances de ces modèles sont leurs hyper-paramètres. Une fois qu’on a défini des valeurs appropriées pour ces hyper-paramètres, les performances d’un modèle peuvent s’améliorer considérablement. Pour ce faire, on utilise la méthode de « GridSearshCv ».

GridSearchCV est une fonction qui vient dans « scikit-learn » model\_selection package. Cette fonction permet de parcourir en boucle des hyper-paramètres prédéfinis et d’ajuster l’estimateur (modèle) aux jeux d’apprentissage. Donc, à la fin, on peut sélectionner les meilleurs paramètres parmi les hyper-paramètres répertoriés.

Validation croisée (cv) est une méthode de validation d’un modèle d’apprentissage qui est nécessaire pour s’assurer que l’algorithme est bien en capacité d’apprendre correctement sur les données de formations et de généraliser sur les données inconnues, c’est-à-dire la capacité d’un modèle à classer correctement les données qu’il ne connaît pas.

### Réglage de l’hyper paramètre du modèle :

Pipline\_mnb: le modèle(classificateur) pour laquelle on souhaite vérifier les hyper-paramètres

Params\_grid: l’objet dictionnaire qui contient les hyper-paramètres qu’on souhaite essayer.

CV: nombre de validation croisée qu’on doit essayer pour chaque ensemble sélectionné d’hyper-paramètres.

Verbose: on peut le définir sur 1 pour obtenir l’impression détaillée pendant qu’on ajuste les données à GridSearchCV

n\_jobs: nombre de processus qu’on veut exécuter en parallèle pour cette tâche si elle « -1 » il utilisera tous les processeurs disponibles.

### Visualisation des résultats :

TP (*vrais positif*), FP (*faux positif*), FN (*faux négatif*), TN (*vrai négatif*), Macro avg (*la moyenne pondérée*)

#### Matrice de confusion de naïve bayésien :

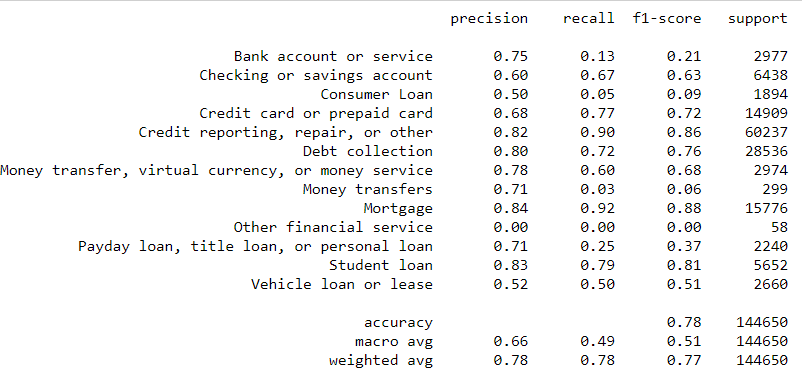
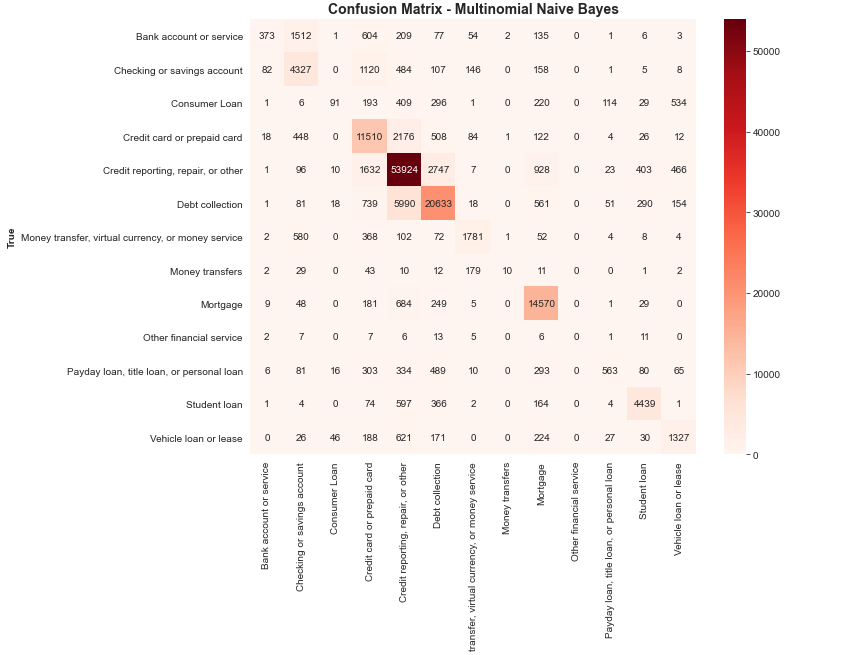


Figure 26 : Visualisation des résultats du classificateur Naïve bayésien



Exemple de calcul pour la ligne 1 (classe : Bank account or service) :

Le classificateur de naïf bayésien a prédit 373 classes dans la classe réelle(correcte) qui revient donc à « 373 de vrais positifs(TP) ». Le classificateur a aussi prédit « 125 de faux positifs(FP) », il a aussi prédit « 2604 de vrais négatifs (TN) ».

Donc pour le calcul de la précision qui est le rapport entre les vrais positifs et le total des vrais positifs et des faux positifs. La précision cherche à voir combien de classes positives ont été trouvées. S’il n’y a pas de mauvais positifs (ces FP), le modèle avait une précision de 100%. Plus il y a de FP, plus la précision va diminuer.

La précision :

La précision examine le nombre de faux positifs qui ont été trouvé dans la combinaison de prédictions. Pour classe « Bank account or service » le calcul de la précision est le suivant :

.

Donc on constate que ce classificateur a 75% de bonne prédiction pour la classe (Banck account or service).

Ce résultat est satisfaisant cela revient aux vrais positifs qui sont plus important que les faux positifs.

Le recall :

Le recall examine le nombre de faux négatifs (les données mal prédites) qui ont été trouvé dans la combinaison de prédictions:

Le taux de recall est pénalisé chaque fois qu’un faux négatif est prédit. Dans cette classe on constate que le rappel est de 13%. Ce résultat est médiocre car le Faux Négatifs sont plus grand que les vrais positifs.

Le F1-score :

Le calcul de f1\_score est la moyenne harmonique entre la précision et le recall:

.

Ce résultat n’est pas satisfaisant ceci dit que le classificateur a mal classé les données. Il arrive à classer correctement seulement 21% des données.

Discussion des résultats du classificateur naïf bayésien :

On constate que la précisions de ce classificateur est de 0,66 et le recall est de 0,49.

Le calcul de F1-score c’est la moyenne des tous les F1-score qui est la moyenne harmonique entre le recall et la précisions et pour cela le classificateur naïf bayésien a donné

**f1-score=0,51**. Ce classificateur donne des résultats satisfaisants.

#### Classificateur support à vecteur machine

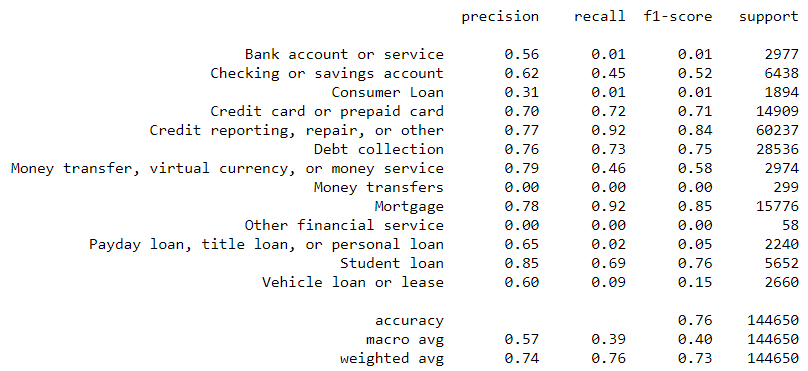
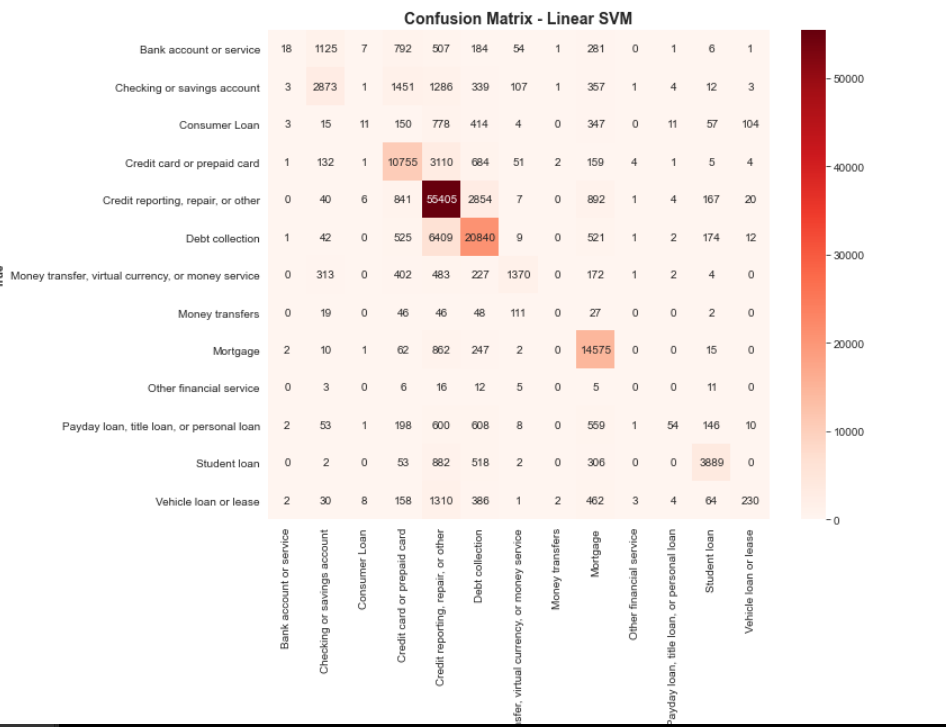


Figure 27 : Visualisation des résultats du support à vecteur machine

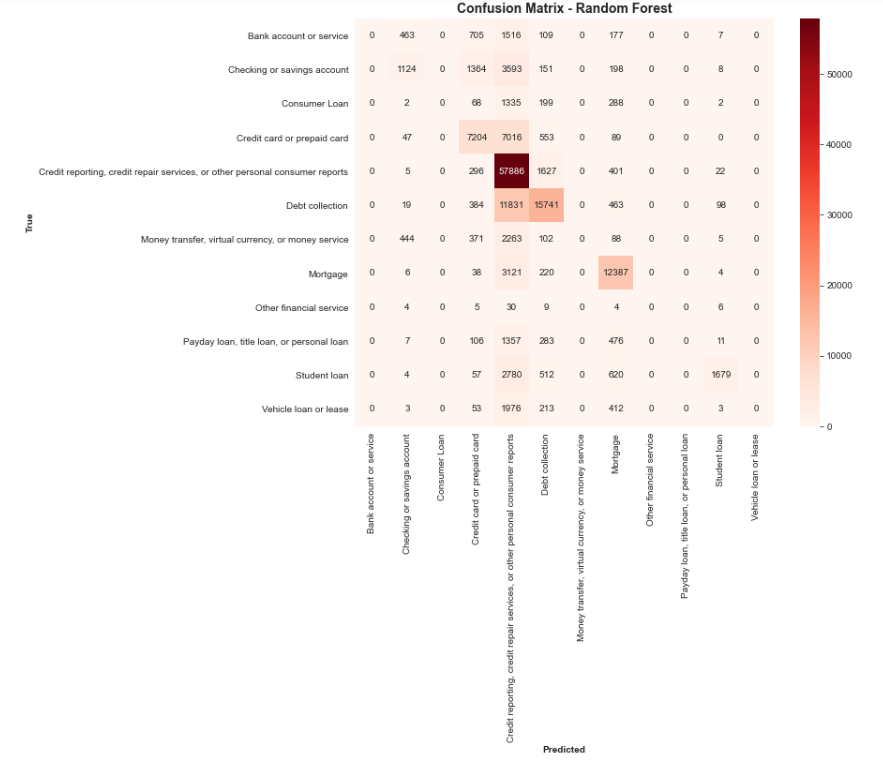


Discussion des résultats du classificateur support à vecteur machine :

Le classificateur support à vecteur machine donne une précision de 0.57 et le recall de 0,39. Cela montre que le classificateur arrive donne uniquement 39% de bonnes prédictions.

Le calcul de macro F1-score est la moyenne des tous les F1-score et pour cela ce classificateur a donné **f1-score=0,40**. Le classificateur support à vecteur machine donne des résultats pas très satisfaisants.

#### Classificateur de forêt aléatoire

****Figure 28 : Visualisation des résultats du classificateur de forêt aléatoire

Discussion des résultats du classificateur de forêt aléatoire :

On constate que la précisions est de 0,36. Ce classificateur donne uniquement 36% de bonne prédiction (Les vrais positifs sont inférieurs aux faux positifs). Le recall est de 0,27.

Le calcul de F1-score de ce classificateur a donné **f1-score=0,29**. Pour conclure, le classificateur de forêt aléatoire donne des résultats très faible.

#### Classificateur régression logistique

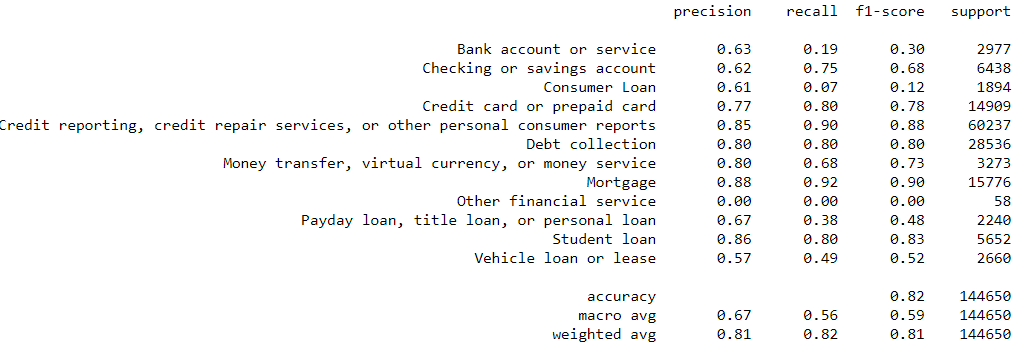


Figure 29 : Visualisation des résultats du classificateur ****régression logistique

Discussion des résultats du classificateur régression logistique :

Le classificateur de régression logistique a donné une précision de 0,67. Cela montre que ce classificateur classe plus que la moitié des données dans les bonnes catégories. Le recall est de 0,56.

Le f1-score de ce classificateur est **f1-score=0,59**. Ce classificateur donne de bonnes précisions. Ces résultats sont satisfaisants.

### Comparaison de différents classificateurs :

|  |  |  |
| --- | --- | --- |
| Modèle de classification | Score | Moyenne Macro-F1 |
| SVM(support à Victor machine) | 0,78 | 0,51 |
| Naïve bayésien | 0,76 | 0,40 |
| Régression logistique | 0,81 | 0,59 |
| La foret aléatoire | 0,66 | 0,29 |

Tableau 6 : Résultat obtenue avec différents classificateurs

Figure 30 : Histogramme présentant la moyenne Macro F1-score des différents classificateurs

### Conclusion du text mining :

À première vue, on remarque que la régression logistique (LR) se distingue et occupe la première place par sa performance. On peut observer que pour la plupart des classes la régression logistique a donné les meilleures performances dans 10 classes sur 12. C’est également LR qui a donné la meilleure valeur de moyenne **macro-F1** qui est de **0,59.**

## Travaux realisé pour le Chatbot

### Chatbot

Un Chatbot est un logiciel intelligent qui est capable de communiquer et d’effectuer des actions similaires à un humain. Il existe deux types de base de modèles Chatbot en fonction de la façon dont ils sont construits: modèle basé sur la récupération et modèle basé sur la génération Dans ce projet on va se baser sur le modèle de récupération.

### Chatbot basé sur la récupération

On construit un Chatbot en utilisant des techniques d’apprentissage profond. Le Chatbot sera formé sur l’ensemble de données qui contient des catégories (intentions), des motifs et des réponses. On utilise un réseau neuronal récurrent spécial (LSTM) pour classer le message de l’utilisateur dans la catégorie appropriée, puis on donnera une réponse aléatoire à partir de la liste des réponses.

On crée un Chatbot basé sur la récupération à l’aide de NLTK, Keras, Python, etc. Un Chatbot basé sur la récupération utilise des modèles d’entrée et des réponses prédéfinis. Il utilise ensuite un certain type d’approche heuristique pour sélectionner la réponse appropriée.

### Importer et charger le fichier de données

Le fichier de données est en format JSON, donc on utilise le paquet python « json »

Pour Analyser le fichier JSON dans Python. Le fichier de question/réponse ressemble à ceci:

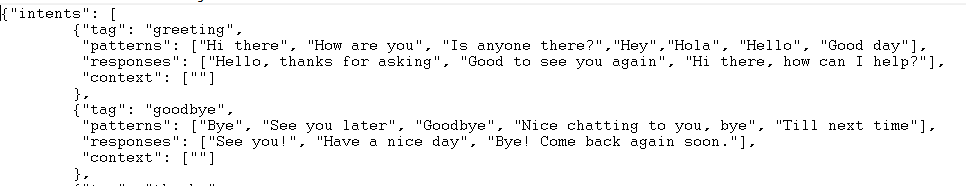


Figure 31 : fichier de question/réponse

### Le prétraitement des données

Lorsqu’on travaille avec des données textuelles, on doit effectuer divers prétraitements sur les données avant de faire un apprentissage automatique.

La tokenization est la chose la plus fondamentale et la première qu’on peut faire sur les données textuelles. La tokenization est le processus de rupture de l’ensemble du texte en petites parties comme les mots.

On réalise des itérations sur les modèles et on symbolique la phrase en utilisant la fonction « nltk.word\_tokenize () » et ajouter chaque mot dans la liste des mots. On crée également une liste de classes pour les balises.

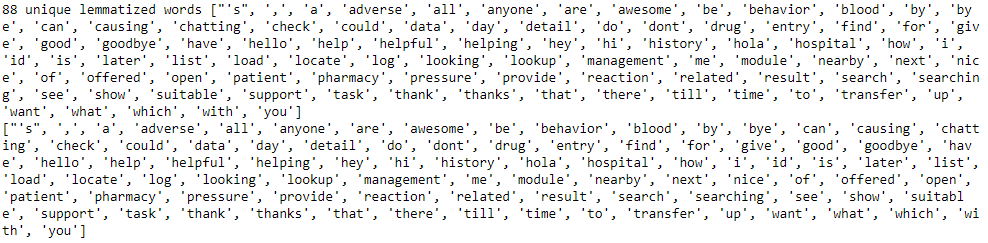


Figure 32 : Exemple de prétraitement des données

### Création des données de formation et de test

On crée les données de formation dans lesquelles on va fournir l’entrée et la sortie. L’entrée sera le modèle et la sortie sera la classe à laquelle le modèle d’entrée appartient. Mais l’ordinateur ne comprend pas le texte, donc on va convertir le texte en nombres en utilisant les sacs de mot (bag of Word ou BOW).

Fonctionnement du Bag of Word :

Le modèle du sac de mots (BOW) est une représentation qui transforme le texte arbitraire en vecteurs de longueur fixe en comptant le nombre de fois où chaque mot apparaît. Ce processus est souvent appelé vectorisation. Il s’agit:

* Un lexique de mots connus.
* Une mesure de la présence de mots connus

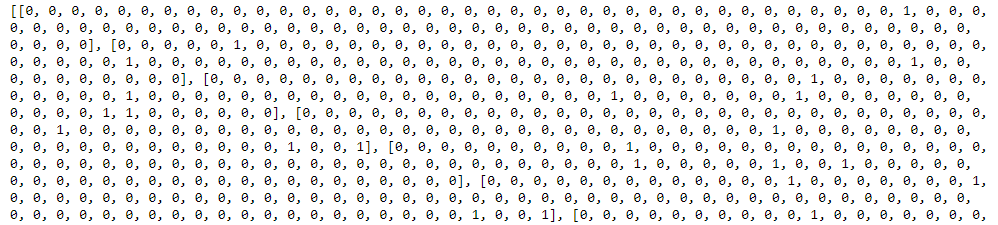
Le résultat du Bag of Word :

Figure 33 : Résultat Bag Of Word

### Construire le modèle d’entrainement pour la classification des intentions

Les données de formation et de test prêtes, on va construire un réseau neuronal séquentiel profond qui possède trois couches.

On crée une instance **Sequential*()*** de la bibliothèque Keras. Celle-ci superpose un ensemble de couches pour en tirer un seul modèle. On lui fournit une liste de couches qu’on souhaite utiliser pour le modèle. La première couche possède 16 neurones, la seconde couche cachée 16 neurones. La couche de sortie contient un nombre de neurones égal au nombre d’intentions à prédire. L’architecture du modèle est la suivante:

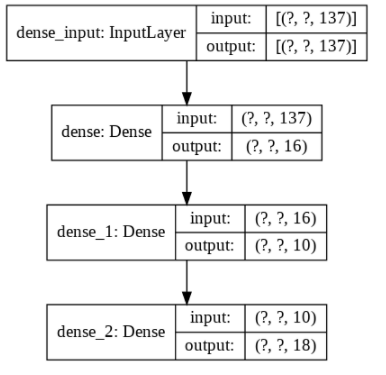
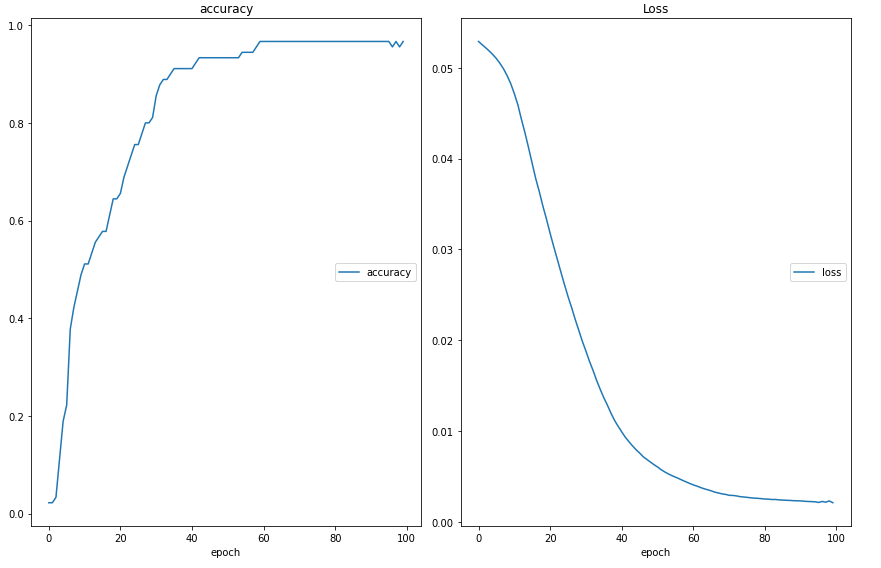


Figure 34 : Architecture du modèle

### Compilation du modèle

L’étape suivante est la compilation du modèle. Pour ce faire, il faut choisir :

* La fonction de perte: utilisée pour diminuer l’erreur. Plus l’erreur est faible plus le modèle est précis ;
* L’optimiseur: aide à obtenir de meilleurs résultats pour la fonction de perte ;
* Métriques accuracy: utilisées pour évaluer le modèle.

Dans le code suivant on a utilisé la fonction de perte comme étant l’erreur quadratique moyenne avec l’optimiseur ***ADAM.*** Et pour la métrique on a utilisé accuracy.

### Résultat obtenue pour l’ensemble d’entrainement

On a entrainé le modèle sur le jeu de données train. Celui-ci passe 100 fois (epochs) sur l’ensemble de données pour en tirer le meilleur résultat possible.

On constate qu’on a atteint une précision de 98,9% avec une fonction de perte de loss=0,00012.

### Evaluation du modèle

On a évalué le modèle sur l’ensemble de test, on a obtenu les résultats suivants:

La prédiction des tests effectués est de 88,6%. Donc on constate que le modèle semble bien fonctionner.

### Exemple de classification

Figure 35 : Exemple de classification de l’intention de l’utilisateur

La fonction classify prend le message de l’utilisateur et classifie la phrase dans sa classe. La prédiction du modèle donne un tableau de probabilités de toutes les classes. D’après la figure ci-dessus on peut voir que la phrase « mot de passe oublié » a une probabilité de 93% qu’elle appartienne à la classe « password reset ».

### Génération de réponse

On charge le modèle formé et on utilise ensuite une interface utilisateur graphique qui prédira la réponse du bot. Le modèle présentera uniquement la classe à laquelle il appartient, donc on va implémenter certaines fonctions qui identifieront la classe et récupéreront ensuite une réponse aléatoire à partir de la liste des réponses.

Encore une fois, on importe les paquets nécessaires et charger les fichiers pickle « words.pkl » et « classes.pkl » qu’on a créés lorsqu’on a formé le modèle:

Pour prédire la classe, On va créer certaines fonctions qui effectueront le prétraitement du texte et ensuite prédire la classe.

* + 1. Visualisation du résultat

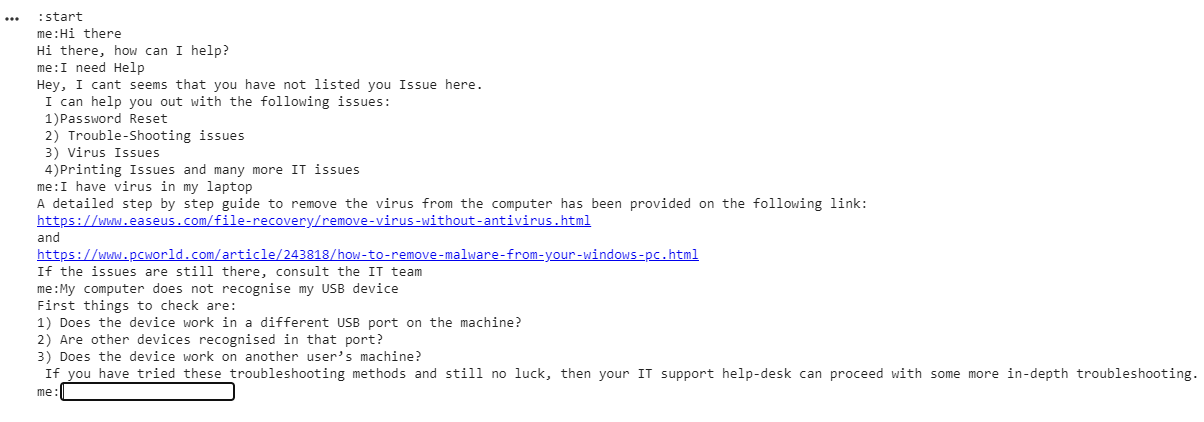


Figure 36 : Génération de réponse

Discussions des résultats :

Le Chatbot ne traite que les termes connus du programme. Lorsqu’on pose une question inconnue au Chatbot, celui-ci ne répondra pas à l’utilisateur, mais il donnera des options à choisir parmi celles qui lui sont familières.

* + 1. Conclusion global du Chatbot

L’application Chatbot semble bien fonctionner pour la base de données à petite échelle. Le Chatbot répondra avec une grande précision si les l’entrée de utilisateurs concernent les sujets mentionnés dans l’ensemble de données. Si on ajoute plus d’information au jeu de données cette application peut être utilisée plus largement que sa capacité actuelle.

# conclusion

Au long de ces travaux de recherche, un système de reconnaissance de voix a été construit en se basant sur les différentes techniques comme le modèle de langage, et le modèle acoustique (cf. sections 6.3.4.6). De plus, à partir des méthodes présélectionnées avec l’étude de l’état de l’art, une sélection plus fine des technologies a pu être effectuée. La première technique est le modèle de n-gram et la deuxième et le modèle Markov caché basé sur les réseaux de neurones dépendant du contexte. Ces technologies correspondent au mieux à la réalisation des tâches (Analyse de la voix humaine).

Au niveau de la mise en œuvre de cet objectif, des difficultés techniques (problème des accents) et un manque de ressources pour l’entrainement ont été constatés. Pour surmonter ces difficultés, on a choisi d’entraîner le modèle par l’utilisation de la méthode du Transfer Learning (L’utilisation des modèles pré-entrainé sur un nouveau jeu de données).

Durant ces travaux, on s’est intéressé aussi à la classification des commentaires des rouleurs.

Cette étude a permis de démontrer que le problème peut être traiter avec une approche basée sur la classification, C’est-à-dire l’ajout d’un label lors d’un signalement de défaut (cf. section 6.4.8).

Le dernier objectif de ces travaux**,** est la réalisation d’un Chatbot pour apporter de l’aide et mieux interagir avec les rouleurs. Pour cela on s’est intéressé à la classification des intentions de l’utilisateurs. Pour cette étude le modèle a été basé sur un réseau de neurones séquentiels (cf. section 6.5.6).

Au niveau de la mise en œuvre du Chatbot, un manque de donnée a été constaté. Pour surmonter cette difficulté, on a choisi de travailler avec des données de sources générales. Néanmoins, Le Chatbot répondra avec une grande précision si les entrées des utilisateurs concernent des phrases qui sont présente dans ce jeu de données d’apprentissage. Il donne une faible précision de réponse dans le cas contraire où l’intention d’entrée ne fait pas partie du domaine du roulage.

# PROGRES SCIENTIFIQUES ET/OU TECHNIQUES

## Valeur ajoutée

Une étape dans la réalisation du projet a consisté à améliorer le système de reconnaissance de la parole déjà développé. Ce système avait un taux d’erreur par mot de 68.6%. Donc l’objectif de départ était d’améliorer l’interface vocale et de la rendre sensible aux accents non-natifs et en mode hors ligne. Plusieurs méthodes de prétraitement ont été expérimentées et vérifiées.

Finalement, une méthode a été retenue pour répondre à chaque problématique (le filtrage du signal, adaptation aux accents non-natifs). Pour cela on a entraîné le modèle qui est basé sur le modèle de DeepSpeech de Mozilla sur un corpus large qui est de 623 h validées.

L’approche qui a été explorée vise à traiter un langage avec un large vocabulaire (LVSCR). Cette approche a donné des résultats satisfaisants par l'amélioration d'un modèle pré-entraîné. On est donc passé d'un modèle avec un taux d’erreurs (WER) de 68 ,6% à un taux d’erreurs (WER) de 54,54%. On peut obtenir de meilleurs résultats si on continue l’entraînement pour plusieurs epochs (nombre de fois que le modèle voit le jeu de données) et en modifiant les hyper-paramètres du décodeur CTC.

Enfin, l’utilisation d’un outil end-to-end et son optimisation ont été explorés, ouvrant la possibilité à son intégration dans l’interface vocale de Diabolo.

Ces travaux ont aussi permis d’étudier plusieurs objectifs comme la classification de texte, qui sert à classer tous les commentaires remontés par les rouleurs selon les catégories appropriées. Plusieurs approches ont été étudiés afin de répondre à cette problématique comme: Naïve Bayes, arbre de décision, support à vecteur machine et la régression logistique. Il s’avère que l’algorithme de la régression logistique a donné de bons résultats de F1-score de 59% et un score de 82%.

Cette étude permet de gagner en temps de saisie, de réduire les risques de pertes d’information et d’avoir la meilleure communication entre les rouleurs et l’équipe de système d’informations.

Enfin, ces travaux ont aussi conduit à la réalisation d’un assistant virtuel. Cela a pour but de mieux interagir avec les rouleurs et leurs apporter de l’aide en temps réel. Parmi les méthodes utilisées, le modèle basé sur les réseaux de neurones semble être celui qui fonctionne le mieux avec une précision de 88,6%.

## Perspectives

Plusieurs perspectives ressortent à la suite de ces travaux :

• Enrichir les jeux de données :

La méthode explorée (LVCSR) peut obtenir de meilleurs résultats en les entraînant sur de plus grandes quantités de données, en les entraînant sur plusieurs epochs et en utilisant plusieurs hyper-paramètres. Une possibilité pour la suite de ces travaux est d’augmenter la quantité de donnée.

• D’autre façon pour générer des données :

La création de données, l’adaptation de jeux de donnée Open Source ou payants. Une autre voie à explorer est la création d’un jeu de données spécifiques en utilisant une plateforme. Cela aurait comme avantage de couvrir un large panel de locuteurs et d’obtenir, sans pertes de temps conséquentes, une grande quantité de données.

• Utiliser des machines puissantes :

Avoir un GPU compatible (NVIDIA, au moins 8 Go de VRAM), il est fortement recommandé d'installer TensorFlow avec prise en charge du GPU. La formation sera beaucoup plus rapide que l'utilisation du processeur.

• Utiliser un autre optimiseur :

Un axe possible d’amélioration est l’utilisation d’autre optimiseur comme RMSprop comme optimiseur.

* Jeu de données métier (classification de texte) :

On a traité un seul corpus pour la classification de texte. Il serait intéressant de tester les approches sur d’autres corpus de textes dans des domaines de l’automobile, et examiner d’autres types de documents. Et enrichir la base de données pour avoir des résultats beaucoup plus performant.

* Utilisation d’autre classificateur comme les K plus proches voisins (KNN)

La méthode des kNN a l’avantage d’être très simple à mettre en œuvre et d’utiliser directement l’ensemble d’apprentissage. Elle ne fait aucune hypothèse a priori sur les données et ne tente pas de créer un modèle pour représenter les classes. La qualité de la discrimination par cette méthode dépend du choix du nombre k de voisins considérés.

Les données sont simplement emmagasinées en mémoire. Pour prédire la classe d’un nouvel objet, on le compare à ses voisins les plus proches par une mesure de similarité.

* Enrichir la base de données question/réponse :

Pour avoir une meilleure précision on doit avant tout enrichir la base de donnée. Et faire une base de données adaptée au langage métier pour pouvoir l’exploiter dans plusieurs contextes.

* Algorithme de classification d’intention :

Essayer d’autre algorithme pour le classificateur d’intention comme les réseaux de neurones récurrents. Qui semble être le meilleur classificateur d’intention.

# BIBLIOGRAPHIE

[1] GHAZI BOUSELMI, « Contributions à la reconnaissance automatique de la parole non-native. Interface homme-machine » [cs.HC]. Université Henri Poincaré - Nancy I, 2008.

[2] ALKA SINGH, SUREKHA GHANGAS, « Speaker recognition using MFCC and DeltaDelta MFCC and classification using artificial neural network», International Journal of Advance Research in Science and Engineering, Vol N°5 (815-819), 2018

[3] HATON, CERISARA, FOHR, LAPRIE, SMAÏLI, « Reconnaissance automatique de la parole: Du signal à son interprétation », Dunod, 2006

[4] HAZEM, ZOUHIR, OUNI, « A study of speech recognition system based on the Hidden Markov Model with Gaussian Mixture », University of Carthage – Tunisia, 2015

[5] L.BAC, “Reconnaissance automatique de la parole pour des langues peu dotées”, Thèse de doctorat, Université Joseph-Fourier-Grenoble I, 2006.

[6] WARD, « Hidden Markov Model in Speech Recognition », Carnegie Mellon University – Pittsburgh

[7] MOHAMED, DAHL, HINTON, « Acoustic modeling using deep belief networks », Audio, Speech and Language Processing, IEEE Transactions on, 2011

[8] SAINATH, KINGSBURY, RAMABHADRAN, FOUSEK, NOVAL, MOHAMED, « Making deep belief networks effective for large vocabulary continuous speech recognition » ASRU, 2011

[9] LI, SEIDE, YU, « Conversationnal speech transcription using context-dependent deep neural networks », Interspeech, 2011

[10] ABDEL-HAMID, MOHAMED, JIANG, PENN, « Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition », Acoustics, Speech and Signal Processing, 2012

[11] ALIBABA CLOUDER, « Interspeech 2017 Series Acoustic Model for Speech Recognition Technology », Février 2018

[12] SAINATH, MOHAMED, KINGSBURY, RAMABHADRAN, « Deep Convolutional Neural Networks for LVCSR », 2013

[13] SAK, SENIOR, BEAUFAYS, « Long Short-Term Memory Recurrent Neural Network Architecture for Large Scale Acoustic Modeling » Interspeech 2014, 2014

[14] SAK, VINYALS, HEIGOLD, SENIOR, MCDERMOTT, MONGA, MAO, « Sequence discriminative distributed training of long short-term memory recurrent network », Interspeech, 2014

[15] SAK, SENIOR, RAO, BEAUFAYS, « Fast and Accurate Neural Networks Acoustic Model for Speech Recognition », 2015

[16] GRAVES, JAITLY, « Towards End-to-End Speech Recognition with Recurrent Neural Networks », ICML, 2014

[17] JYOTHI « Automatic Speech Recognition (CS753) Lecture 1: Introduction to Statistical Speech Recognition », July - 2017

[18] MARIANI, « Reconnaissance automatique de la parole: progrès et tendances », LIMIS-CNRS, 1990

[19] VAUFREYDAZ, « Modélisation statistique du langage à partir d’internet pour la reconnaissance automatique de la parole continue », thèse de doctorat à l’Université J. Fourier – Grenoble, 2002

[20] HANNUN, CASE, CASPER CATANZARO, DIAMOS, ELSEN, PRENGER, SATHEESH, SENGUPTA, COATES, Y.NG, « Deep Speech: Scaling up end-to-end speech recognition », Baidu Research – Silicon Valley AI Lab, 2014

[21] PANAYOTOV, CHEN, POVEY, KHUDANPUR, « LibriSpeech: an ASR corpus based on public domain audio books », ICASSP, 2015

[22][https: //blog.mozilla.org/press-fr/2019/02/28/common-voice-mutualiser-nos-voix-mozilla-publie-le-plus-grand-jeu-de-donnees-vocales-transcrites-du-domaine-public-a-ce-jour/](https://blog.mozilla.org/press-fr/2019/02/28/common-voice-mutualiser-nos-voix-mozilla-publie-le-plus-grand-jeu-de-donnees-vocales-transcrites-du-domaine-public-a-ce-jour/), Explication du principe de la plateforme CommonVoice, 25 juillet 2019

[23] HEAFIELD, « KenLM: Faster and Smaller Language Model Queries », WNT at EMNLP – Edimbourg, 2011

[24] HEAFIELD, POUZYREVSKY, CLARK, KOEHN, « Scalable Modified Kneser-Ney Language Model Estimation », ACL – Sofia, 2013

[25] VITERBI, « Error bounds for convolutional codes and an asymptotically optimum decoding algorithm », IEEE – Transactions on Information Theory, 1967

[26] GRAVES, « Supervised Sequence Labelling with Recurrent Neural Network », Phd Thesis – Technical University Munich, 2008

[27] KINGMA, BA, « Adam: A method for stochastic optimisation », International Conference for Learning Representation – San Diego, 2015

[28] SUTSKEVER, MARTENS, DAHL, HINTON, « On the importance of initialization and momentum in deep learning », ICML, 2013

[29] L. LIU, JIANG, HE, CHEN, X. LIU, GAO, HAN, « On the Variance of the Adaptive Learning Rate and Beyond », 2019

[30] Capo-chichi\_Grace\_2012\_memoire«Apprentissage Automatique pour la détection de relations d’affaire»

[31] Luiza Orosanu « Reconnaissance de la parole pour l’aide à la communication pour les sourds et malentendants » « HAL Id: tel-01251128 [https: //hal.inria.fr/tel-01251128»](https://hal.inria.fr/tel-01251128), 5 Jan 2016

[32] Sébastien DEMANGE « Contributions à la reconnaissance automatique de la parole avec données manquantes » HAL Id: tel-01748268 [https: //tel.archives-ouvertes.fr/tel-01748268v3](https://tel.archives-ouvertes.fr/tel-01748268v3) , 5 Feb 2008

[33] Hussein T. Al-Natsheh « Text Mining Approaches for Semantic Similarity Exploration and Metadata Enrichment of Scientific Digital Libraries » HAL Id: tel-02065269 [https: //hal.archives-ouvertes.fr/tel-02065269](https://hal.archives-ouvertes.fr/tel-02065269), 12 Mar 2019

[34] Adeline ABBÉ « Analyse de données textuelles d’un forum médical pour évaluer le ressenti exprimé par les internautes au sujet des antidépresseurs et des anxiolytiques » HAL Id: tel-01410526 [https: //tel.archives-ouvertes.fr/tel-01410526 2016](https://tel.archives-ouvertes.fr/tel-01410526%202016)

[35] Peters, Florian « Design and implementation of a chatbot in the context of customer support ». 2018

[36] DANESI Charlot « Adaptation des techniques de Text Mining aux données Conversationnelles issues de l’oral » HAL Id: dumas-00567888 [https: //dumas.ccsd.cnrs.fr/dumas-00567888](https://dumas.ccsd.cnrs.fr/dumas-00567888) , 22 Feb 2011

[37] Bouziane Abdelghani « Exploitation des données liéess: Système Question-Réponse »2019

[38] G. Cybenko. « Approximation by superpositions of a sigmoidal function ». In:

Mathematics of Control, Signals and Systems 2 (1989), p. 303-314.

[39] Kurt Hornik, Maxwell Stinchcombe et Halbert White. « Multilayer feedforward

networks are universal approximators ». In: Neural Networks 2.5 (1989), p. 359-366.

issn: 0893-6080. doi: [https: //doi.org/10.1016/0893-6080(89)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).

url:https: //www.sciencedirect.com/science/article/pii/0893608089900208.

[40] Jack Cahn « CHATBOT: Architecture, Design, & Development » April 26, 2017

# Annexe

|  |  |  |
| --- | --- | --- |
| Wav\_filename | Transcript\_input | Transcript\_output |
| Fr-2026413008 | Un tel écart nécessitait forcément le recours à un troisième moteur | Un tel car laissait fortement le recours a trois ie moteur |
| Fr-18023605 | Cela me gêne car une telle rédaction est un peu floue | Cela me jeune par un telle relation est en pochon |
| Fr-20475874 | La structure de la dynactine est très conservée chez les vertébrés | La structure de la inactive est pris conserve sur le vertige |
| Fr-18576591 | Il vous donne la même raison de choisir notre devoir | Il vous de la même raison de faire notre de voir |

## Annexe1 : Résultats du test de la RAP

## Annexe 2 : Code calcul du WER et CER

def process\_decode\_result(item):

label, decoding, distance, loss = item

word\_distance = levenshtein(label.split(), decoding.split())

word\_length = float(len(label.split()))

return AttrDict({

'src': label,

'res': decoding,

'loss': loss,

'distance': distance,

'wer': word\_distance / word\_length,

})

def calculate\_report(labels, decodings, distances, losses):

# This routine will calculate a WER report.

# It'll compute the mean WER and create ``Sample`` objects of

# the ``report\_count`` top lowest

# loss items from the provided WER results tuple (only items with WER!=0

# and ordered by their WER).

samples = pmap(process\_decode\_result, zip(labels, decodings,

distances, losses))

# Getting the WER and CER from the accumulated edit distances and lengths

samples\_wer, samples\_cer = wer\_cer\_batch(labels, decodings)

# Order the remaining items by their loss (lowest loss on top)

samples.sort(key=lambda s: s.loss)

# Then order by WER (highest WER on top)

samples.sort(key=lambda s: s.wer, reverse=True)

return samples\_wer, samples\_cer, samples

## Annexe 3 : Code distance de Levenshtein

# The following code is from:

# http: //hetland.org/coding/python/levenshtein.py

# This is a straightforward implementation of a well-known algorithm, and thus

# probably shouldn't be covered by copyright to begin with. But in case it is,

# the author (Magnus Lie Hetland) has, to the extent possible under law,

# dedicated all copyright and related and neighboring rights to this software

# to the public domain worldwide, by distributing it under the CC0 license,

# version 1.0. This software is distributed without any warranty. For more

# information, see <http: //creativecommons.org/publicdomain/zero/1.0>

def levenshtein(a, b):

"Calculates the Levenshtein distance between a and b."

n, m = len(a), len(b)

if n > m:

# Make sure n <= m, to use O(min(n,m)) space

a, b = b, a

n, m = m, n

current = list(range(n+1))

for i in range(1, m+1):

previous, current = current, [i]+[0]\*n

for j in range(1, n+1):

add, delete = previous[j]+1, current[j-1]+1

change = previous[j-1]

if a[j-1] != b[i-1]:

change = change + 1

current[j] = min(add, delete, change)

return current[n]