Université Paris Dauphine-PSL

Projet IA sur le Cloud - Cas 4 - NLP

Auteurs :
Mehdi Brahmi
Elise Chin
Mathilde Da Cruz
Mohamed Rahmouni

April 8, 2022

1 Formulation du problème

"Beaucoup de mes clients déposent leur voiture après les heures d'ouverture, mettent les clés dans notre boîte aux lettres et laissent un message vocal pour nous dire quel est le problème. Taper le message vocal sur l'ordinateur prend trop de temps et je veux m'assurer que mes techniciens ont accès à ces informations au format texte sur leurs tablettes."

Nous pouvons décomposer notre problème en 2 sous-problèmes successifs :

- Reconnaissance vocale
- Extraction de mots-clés

2 Analyse de l'existant sur AWS

Sur AWS, il existe un module pour chaque partie de notre problème :

- Reconnaissance vocale: Amazon Transcribe https://aws.amazon.com/fr/transcribe/
- Extraction de mots-clés: Amazon Comprehend https://aws.amazon.com/fr/comprehend/

Il existe également un module qui permettrait de répondre à notre problématique : Amazon Transcribe Call Analytics .
Il s'agit d'une API qui, à partir de la reconnaissance vocale d'un appel, identifie les parties clés de la conversation avec le client, leur attribue une étiquette puis enregistre les éléments de synthèse de l'appel dans un fichier de sortie sur S3.
https://aws.amazon.com/transcribe/call-analytics/

3 SOTA

3.1 Reconnaissance vocale automatique

3.1.1 Reconnaissance vocale - Définition

Soit X un signal audio d'un enregistrement vocal. On cherche f qui transforme X en une séquence des mots prononcés.

$$W = f(X)$$

- Les caractéristiques X sont soit le signal audio ou d'autres représentations obtenues à partir de transformations temporelles et/ou spectrale.
- Concernant la sortie W, la plupart des modèles considère un problème intermédiaire W' = g(X) où W' est une séquence de phonèmes. Un phonème est la plus petite unité discrète ou distinctive que l'on puisse isoler par segmentation dans la chaîne parlée. Exemple: AA, AY...

3.1.2 Reconnaissance vocale - Métriques d'évaluation

Pour pouvoir comparer les modèles de reconnaissance vocale, nous allons nous baser sur une métrique de performance du modèle. Il existe plusieurs métriques d'évaluation en reconnaissance vocale. La plus

utilisée est le **taux d'erreur de mots** ou **"Word Error Rate" (WER).**

$$\text{WER} = \frac{S+D+I}{N} \text{ où } \begin{cases} S = \text{Nombre de mots remplacés} \\ D = \text{Nombre de mots supprimés} \\ I = \text{Nombre de mots ajoutés} \\ N = \text{Nombre de mots prononcés} \end{cases}$$

Exemples :
(S) "I surf small waves" → "I surf **all** waves"
(D) "I surf small waves" → "I surf waves"
(I) "I surf waves" → "I surf **small** waves"

Il existe plusieurs approches pour résoudre un problème de reconnaissance vocale. Nous allons considérer l'approche statistique et l'approche avec apprentissage profond qui sont les plus populaires.

3.1.3 Reconnaissance vocale - Approche statistique

L'approche statistique consiste à trouver la séquence de mots W la plus probable étant donné une séquence audio d'entrée X.

Le problème consiste à maximiser la probabilité:

$$W^* = \text{argmax } P(W|X)$$

Avec le théorème de Bayes, le problème revient à maximiser la probabilité :

$$W^* = \text{argmax } P(X|W)P(W)$$

où $P(X)$ est le modèle de langue (probabilité d'apparition d'un mot) et $P(X|W)$ est la fonction de vraisemblance ou le modèle acoustique.

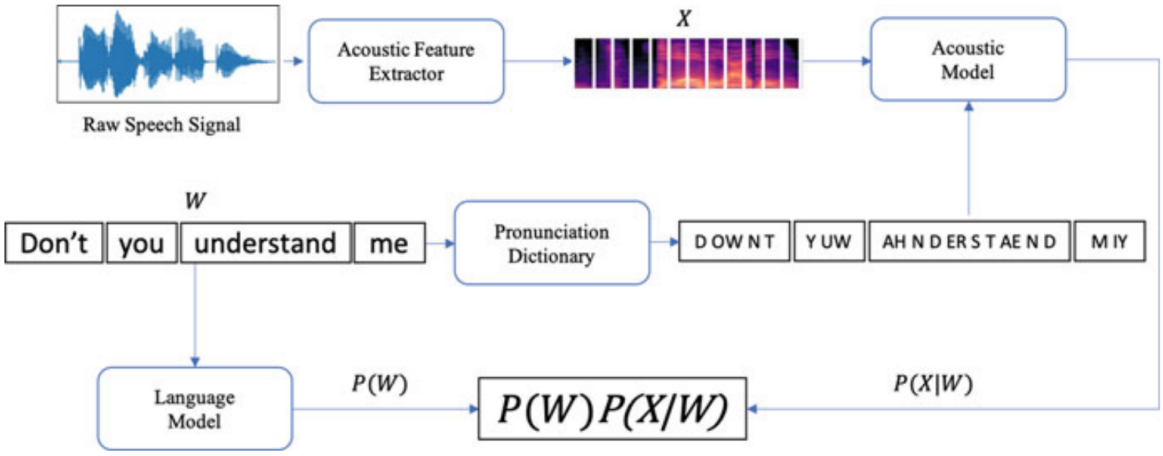

Figure 1: Exemple de modèle statistique de reconnaissance vocale automatique

3.1.4 Reconnaissance vocale - Approche avec apprentissage profond

Les modèles d'apprentissage profond (bout-en-bout) les plus récents consistent pour la plupart à apprendre une représentation vectorielle du signal audio pour résoudre le problème de reconnaissance vocale.

L'un des défis de la reconnaissance vocale est que l'entraînement de réseaux neuronaux requiert un alignement entre les phonèmes et le signal audio. En général, il y a une différence significative entre le nombre d'échantillons du signal audio T et la taille de la séquence de mots N.
Exemple : supposons qu'on utilise des fenêtres glissantes de 10 ms pour générer des prédictions sur le signal audio et qu'une personne prononce 10 mots en 3s. Il en résulte une séquence de taille de 300 en entrée avec une séquence

de mots de taille 10 en sortie.
Construire manuellement l'alignement audio-phonème s'avère très coûteux pour de grands jeux de données. On souhaiterait donc faire correspondre une séquence de mots avec un signal audio **sans avoir à aligner chaque phonèmes à chaque fenêtre du signal audio**.
Une première solution simple serait de combiner les termes répétitifs en un caractère.

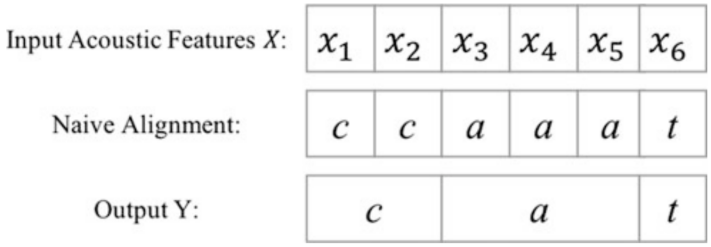

Figure 2: Exemple de stratégie d'alignement du signal d'entrée X avec le mot "cat" en sortie

Cette solution reste très limitée pour deux raisons :

- En reconnaissance vocale, le signal audio peut contenir des périodes de silence qui ne sont pas alignés avec le texte.
- Avec cette solution, on ne peut pas représenter les mots avec des caractères répétitifs.

Une solution très utilisée est l'algorithme de **classification temporelle connexionniste (CTC)**. Cet algorithme impose au réseau de prédire un caractère par fenêtre du signal audio et ajoute un caractère spécial ϵ appelé "non-caractère" ou "blank".

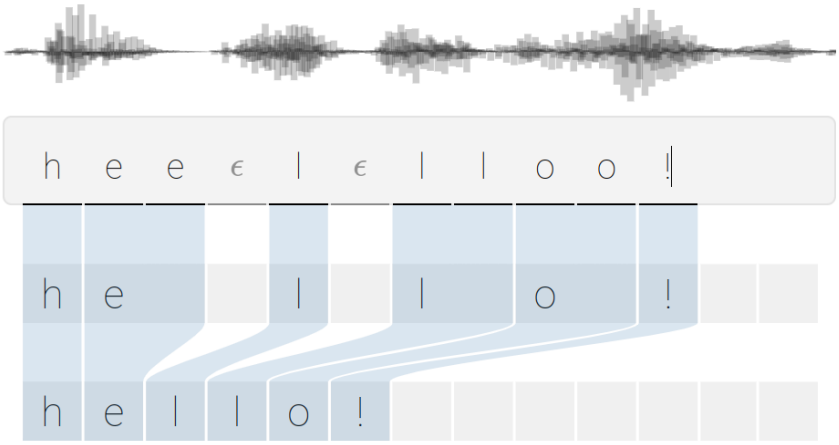

Figure 3: Schéma de l'algorithme classification temporelle connexionniste (CTC)

L'algorithme CTC est utilisé au niveau de la couche de sortie d'un réseau de neurones récurrent, qui permet d'entraîner celui-ci avec des séquences de labels différents, au lieu d'affecter un même label à tous les instants comme dans la Figure 2.

L'utilisation de cette sortie permet de simplifier l'apprentissage du modèle car elle permet de s'affranchir de l'annotation des instants d'apparition de chaque caractères. Concrètement, CTC permet, à partir des sorties du réseau (sur chaque fenêtre temporelle) et de la séquence de caractères cible, de calculer les séquences de vecteurs d'erreurs à rétro-propager.
Le fonctionnement détaillé de la sortie CTC est décrit dans [3].

3.1.5 Évolution de l'état de l'art de la reconnaissance vocale automatique.

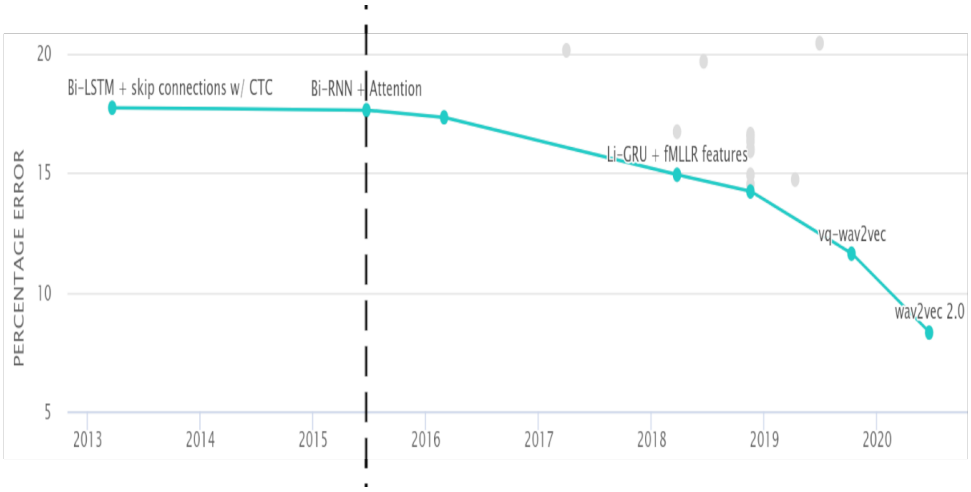

Figure 4: Évolution des performances des modèles état de l'art en reconnaissance vocale automatique. Benchmark réalisé sur TIMIT

TIMIT est un corpus de discours transcrits phonémiquement et lexicalement par des anglophones américains de différents sexes et dialectes (326 voix d'hommes and 136 voix de femmes).

La plupart des réseaux de neurones les plus récents sont basés sur l'utilisation de réseaux de mémoire bidirectionnelle à long terme déroulés sur l'ensemble de l'énoncé de la parole afin de capturer plus de contexte acoustique à chaque fenêtre temporelle du signal audio. En sortie, une couche de sortie CTC est utilisée en tant que fonction de coût.
Les mécanismes d'attention ont commencé à être utilisés entre 2015 et 2016.

3.1.6 Deep Speech : Modèle bi-RNN avec CTC (2014)

Avant 2015, les réseaux les plus performants se caractérisaient par des couches LSTM bi-directionnelles avec une couche de sortie CTC. Comme exemple, on peut citer le modèle Deep Speech publié en 2014 [4].

En entrée, le spectrogramme du signal signal est utilisé. La couche d'entrée est ensuite suivie de couches denses. Au vu de la complexité de la structure bout-en-bout, le réseau nécessite un jeux de données conséquent pour donner de bons résultats. A sa sortie Deep search, Deep Search avait été entraîne sur 5000 heures de séquence audio provenant de 9600 voix différentes.

Une version améliorée, Deep Speech 2[1], a été développée en 2015. La nouvelle architecture a permis de réduite la durée d'entraînement par un facteur 7 par rapport à l'implémentation Deep Speech.

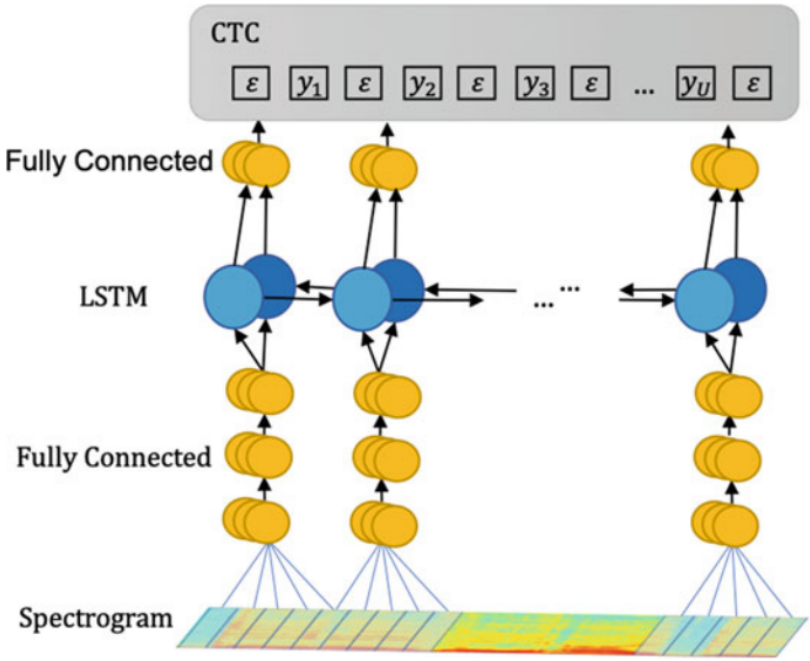

Figure 5: Modèle Deep Speech

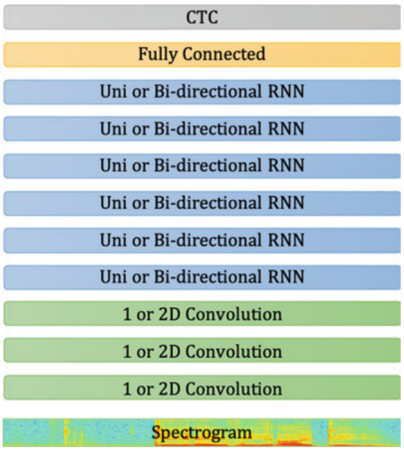

Figure 6: Modèle Deep Speech 2.0

Cette nouvelle architecture introduit des couches de convolution à 1D ou 2D appliquées sur le spectrogramme d'entrée.

3.1.7 Modèles avec attention

A partir de 2015, les modèles de séquence ont été améliorés à l'aide d'un mécanisme d'attention. Cet algorithme aide le modèle à comprendre où celui-ci doit focaliser son attention, compte tenu d'une séquence d'entrées. Dans cette section, nous expliquerons le principe d'attention de façon globale, mais celui-ci peut-être étendu à plusieurs domaines : Computer Vision, NLP et notamment la reconnaissance vocale (ce qui nous intéresse ici).

Les Encodeurs-Décodeurs permettent dans la plupart des cas de transformer une séquence en une autre. Néanmoins, le vecteur créé par l'Encodeur est fixe et cela pose problème pour transformer de longues séquences. Le mécanisme de l'Attention va, d'une part, déjouer ce problème, et d'autre part améliorer les capacités de l'Encodeur-Décodeur.

L'Attention comment ça marche ?

L'approche classique est de prendre la sortie de l'Encodeur pour ensuite la transmettre au Décodeur qui va nous donner le résultat de notre modèle. Avec le mécanisme de l'Attention, au lieu de se concentrer uniquement sur la sortie finale du RNN, l'Attention va prélever des informations lors de chacune des étapes du RNN

Dans une couche RNN classique on a une sortie pour chaque mot. Chaque sortie sera utilisé pour calculer la sortie du mot suivant et ainsi de suite. C'est la récurrence. Dans une couche RNN avec Attention, on a aussi ce calcul récurrent sur chaque mot. Mais en plus de cela, on garde chacune de ces sorties récurrentes en mémoire pour former la sortie finale. Dit autrement, l'Encodeur de l'Attention va transmettre beaucoup plus d'informations au Décodeur que lors de l'approche classique.

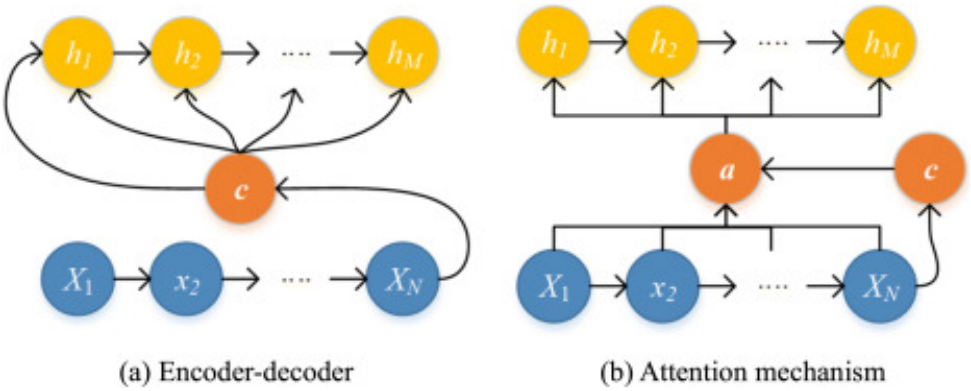

Figure 7: Mécanisme d'attention vs RNN

Le Décodeur de l'Attention utilise, lui aussi, un RNN mais il ajoute ensuite d'autres calculs plus complexes. Sa fonction principale est de comprendre les relations entre chacun des vecteurs d'entrée (par exemple des mots dans la traduction). Cette relation va nous permettre d'identifier les liens qui unissent les mots entre eux. Ainsi notre modèle va comprendre quel verbe se rapporte à quel sujet, quel sujet est associé à quel adjectif.

l'Attention permet à deux vecteurs, même très éloignés dans la séquence d'entrée, de comprendre leur relation. C'est tout l'avantage du mécanisme de l'Attention. Lors de l'entraînement des réseaux de neurones, le modèle focalise son attention sur chaque vecteur de la séquence. Ainsi le modèle peut détecter le contexte des vecteurs et avoir une compréhension globale de la phrase, et donc fournir une meilleure transformation.

3.1.8 Etat de l'art : Wav2vec 2.0

Wav2Vec 2.0 est l'un des modèles de pointe actuels pour la reconnaissance automatique de la parole grâce à un apprentissage auto-supervisé, un concept assez nouveau dans ce domaine. Cette méthode d'apprentissage nous permet de pré-entraîner un modèle sur des données non étiquetées, qui sont toujours plus accessibles. Ensuite, le modèle peut être affiné sur un jeu de données particulier dans un but précis. Comme le montrent les travaux précédents, cette méthode d'apprentissage est très puissante.

L'architecture du modèle final utilisé pour la prédiction se compose de trois parties principales :

- Couches convolutionnelles qui traitent l'entrée de la forme d'onde brute pour obtenir une représentation latente - Z,
- couches de transformation, créant une représentation contextualisée - C,
- projection linéaire vers la sortie - Y.

Voilà à quoi ressemble le modèle après le réglage final, prêt à être développé dans un environnement de production. Toute la magie se produit pendant la première phase de formation, en mode auto-supervisé, lorsque le modèle a un aspect un peu différent. Le modèle est entraîné sans la projection linéaire générant une prédiction de sortie.

Contrastive Learning

Le Contrastive Learning est un concept dans lequel l'entrée est transformée de deux manières différentes. Ensuite, le modèle est entraîné à reconnaître si deux transformations de l'entrée sont toujours le même objet. Dans Wav2Vec 2.0, les couches de transformation sont la première façon de transformer, la seconde est faite par quantification, ce qui sera expliqué après la figure. Plus formellement, pour une représentation latente masquée **Z**, nous aimerions

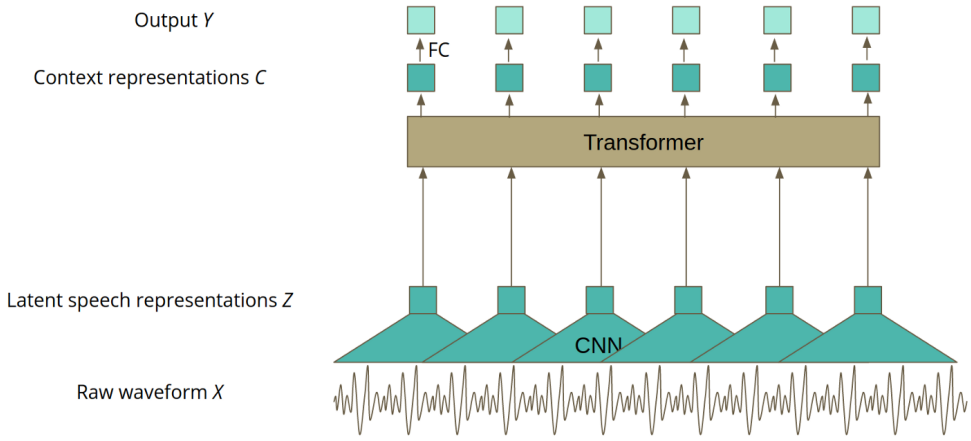

Figure 8: Architecture du Modèle wav2vec 2.0 fine-tuné [2]

obtenir une telle représentation contextuelle **C** pour pouvoir deviner la représentation quantifiée correcte **Q** parmi les autres représentations quantifiées.

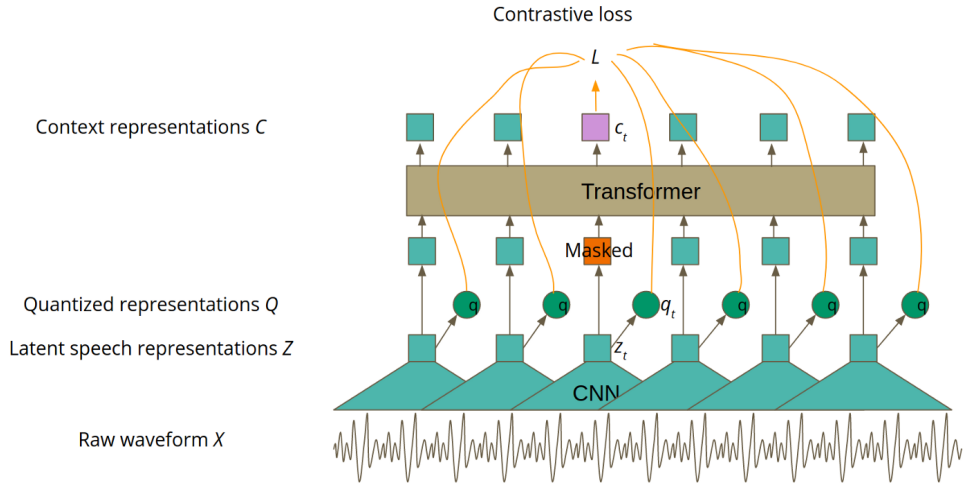

Figure 9: Modèle wav2vec 2.0 [2]

La quantification

La quantification est un processus de conversion des valeurs d'un espace continu en un ensemble fini de valeurs dans un espace discret.

Pour implémenter cela dans la reconnaissance automatique de la parole, nous utilisons un vecteur latent de représentation de la parole **Z** qui couvre deux phonèmes. Le nombre de phonèmes dans une langue est fini. De plus, le nombre de toutes les paires de phonèmes possibles est fini. Cela signifie qu'ils peuvent être parfaitement représentés par la même représentation latente de la parole. De plus, leur nombre est fini, nous pouvons donc créer un livre de code contenant toutes les paires de phonèmes possibles. Ensuite, la quantification se résume à choisir le bon mot de code dans le livre de codes. Cependant, nous pouvons imaginer que le nombre de tous les sons possibles est énorme. Pour faciliter l'entraînement et l'utilisation, les auteurs de Wav2Vec 2.0 ont créé G codebooks, chacun composé de V mots de code. Pour créer une représentation quantifiée, le meilleur mot de chaque livre de codes doit être sélectionné. Ensuite, les vecteurs choisis sont concaténés et traités par une transformation linéaire pour obtenir une représentation quantifiée.

Wav2Vec 2.0 utilise une approche de formation auto-supervisée pour la reconnaissance automatique de la parole, qui est basée sur l'idée de l'apprentissage contrastif. L'apprentissage de la représentation de la parole sur un énorme ensemble de données brutes (non étiquetées) réduit la quantité de données étiquetées nécessaires pour obtenir des résultats satisfaisants.

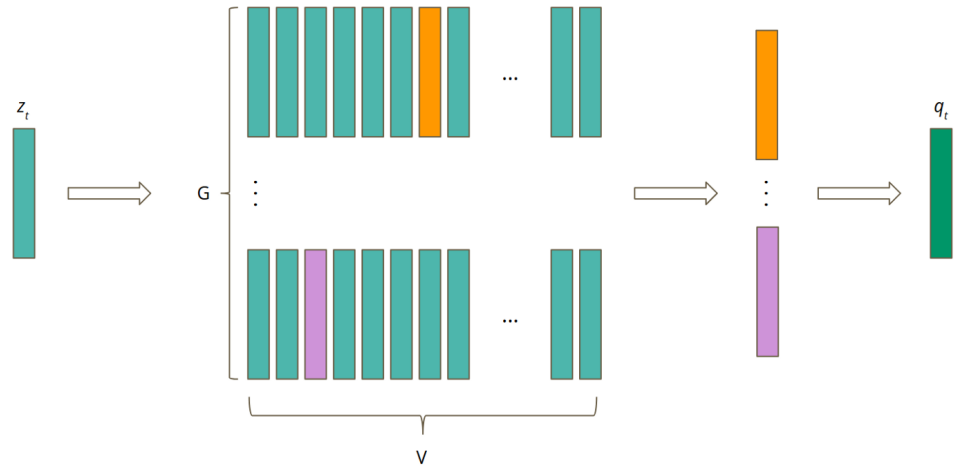

Figure 10: Quantization [2]

3.2 Automatic Keyword Extraction (AKE)

3.2.1 AKE - Définition

L'**extraction de mots-clés** (*Keyword extraction (KE)*) consiste à extraire les mots représentant au mieux un document textuel, l'objectif principal étant de fournir un premier aperçu de son contenu. L'automatisation de cette tâche à l'aide de modèles d'apprentissage ou d'approches statistiques est appelé **extraction automatique de mots-clés** (*Automatic Keyword Extraction (AKE)*).

3.2.2 AKE - Cas non supervisé

Pour notre étude de cas, il s'agira d'une tâche non supervisée, puisque le document texte extrait de l'audio ne sera pas labellisé.

Les méthodes utilisées pour l'extraction non supervisée de mots-clés peuvent être divisées en deux écoles :

- Linguistique : la méthode la plus populaire est d'analyser la distribution des sujets dans un corpus de documents (KeyCluster, CommunityCluster...)
- Statistique : analyse la probabilité des caractéristiques des mots d'un texte (KP-Miner, YAKE, TF-IDF, TextRank, SingleRank...)

3.2.3 AKE - Métriques d'évaluation

Suppose d'avoir des annotations au préalable.

- Statistique :
  - Precision : accuracy des mots-clés extraits par l'algorithme
  - Recall :
  - Fα-score: combinaison de la precision et du recall
- Linguistique :
  - Mean Reciprocal Rank (MRR)
  - Mean Average Precision (MAP)
  - Binary Preference Measure (Bpref)

Evaluation Metrcis	precision	$precison = \frac{tp}{tp+fp} = \frac{thenumberofcorrectkeyphrase}{thenumberofextractedkeyphrase}$
	recall	$recall = \frac{tp}{tp+fn} = \frac{the\ number\ of\ correctly\ matched\ keyphrase}{the\ number\ of\ assigned\ keyphrase}$
	F-socre	$F\alpha - score = \frac{(\alpha^2+1)\cdot precision \cdot recall}{\alpha^2 \cdot precision + recall}$
	MRR	$MRR = \frac{\sum_{d\in D} \frac{1}{rank_d}}{\lvert D\rvert}$
	AP	$AP = \frac{\sum_{n=1}^{\lvert N\rvert} P(n)gd(n)}{\lvert LN\rvert}$
	MAP	$MAP = \frac{1}{n}\sum_{i=1}^{n} AP_i$
	Bpref	$Bpref = \frac{1}{C}\sum_{c\in C} 1 - \frac{\lvert I\rvert}{M}$

Figure 11: Formules des métriques d'évaluation pour l'extraction automatique de mots-clés [5]

3.2.4 AKE - Approches

Les approches non supervisées en extraction de mots-clés peuvent être catégorisées en quatre groupes : approches statistiques, approches basées sur les graphes, approches basées sur les thèmes (*topic-based*) et approches basées sur des modèles de langage. La figure 12 résume ces méthodes.

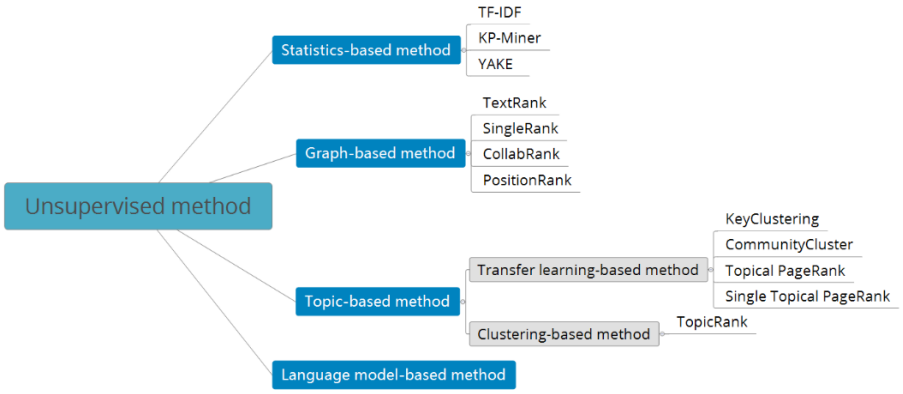

Figure 12: Résumé des méthodes non supervisées pour l'extraction de mots-clés

Approches statistiques. La méthode la plus commune dans le domaine de l'extraction de mots-clés est **TF-IDF** où TF pour *Term Frequency* représente la fréquence d'un mot dans un document et IDF pour *Inverse Document Frequency* représente le nombre de documents où un mot T apparaît. Cette mesure permet d'évaluer l'importance d'un terme dans un document relativement aux autres documents d'un corpus, c'est le cas lorsque TF et IDF sont grands. En 2009, une autre méthode statistique, **KP-miner**, est proposée et se découpe en trois étapes : la première consiste à sélectionner des candidats à l'aide de caractéristiques statistiques, la deuxième calcule le score TF-IDF pour chacun des candidats et la dernière sélectionne les candidats avec le plus haut score, qui seront considérés comme les mots-clés finaux. **YAKE** est introduit en 2018 et comme KP-miner repose également sur TF-IDF, la différence étant l'utilisation d'un nouveau ensemble de caractéristiques pour la sélection des candidats. L'ensemble de ces méthodes sont appliquées sur un corpus de texte, possiblement longs, et ne semblent donc pas adapté à notre étude de cas. En effet, les textes extraits de messages vocaux sont non seulement courts mais nous souhaitons aussi les traiter individuellement pour y extraire les problèmes principaux soulevés dans le message.

Approches basées sur les graphes. Dans cette approche, la tâche d'extraction de mots-clés est reformulée en un problème d'ordonnancement sur des graphes en supposant que plus un mot (noeud) aura de connexions (arêtes), plus il sera un candidat important. Cette idée se base sur celle de PageRank de Google où chaque arête est considérée comme un vote, donc plus un noeud reçoit de votes, plus son score sera élevé. **TextRank** (2004) est le premier algorithme à employer PageRank pour l'extraction de mots-clés. Un graphé orienté est généré de la manière suivante : un noeud représente un mot, et une arête est placée entre deux mots pour représenter leur lien de co-occurrence dans une fenêtre fixée au préalable. PageRank est ensuite exécuté sur ce graphe pour calculer le score final de chaque noeud. **SingleRank** (2008) introduit des poids sur les arêtes représentant le nombre de fois où deux mots apparaissent dans la même fenêtre simultanément. Les poids permettraient de refléter la relation sémantique entre deux noeuds. Enfin, **PositionRank** (2017) ajoute dans SingleRank la localisation des mots dans le texte. L'algorithme exploite l'idée que plus un candidat apparaît suffisamment tôt dans le texte, plus il sera important.

Approches basées sur les thèmes. Nous pouvons supposer qu'un message vocal ne porte que sur un seul sujet (description du problème) et que ces méthodes sont pertinentes pour notre cas d'étude.

Approches basées sur les modèles de langage. **KeyBERT**[1] exploite les embeddings de BERT pour créer des mots-clés similaires à un document. Tout d'abord, un embedding du document est extrait à l'aide d'un modèle BERT pré-entraîné spécifique au domaine. Ensuite, les embeddings de mots sont extraits pour des N-gram. Enfin, KeyBERT utilise la similarité cosinus comme mesure de similarité pour trouver les mots/phrases qui sont les plus similaires au document original. Les mots top-k correspondent à l'ensemble final de mots-clés.

4 Solution custom

Nous envisageons, dans un premier temps, d'utiliser un modèle wav2vec 2.0 pré-entraîné sur des corpus en français. Certains modèles sont disponibles sur Hugging Face comme **wav2vec2-large-xlsr-53-french**.

Concernant l'extraction des mots-clés, nous allons commencer par tester les performances des **approches basées sur les graphes**. En fonction des résultats, nous pourrons ensuite passer à des modèles plus complexes à base de transformers comme **KeyBERT**.

[1] https://github.com/MaartenGr/KeyBERT

References

[1] Dario Amodei et al. "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin". In: Dec. 2015.

[2] Alexei Baevski et al. "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In: *CoRR* abs/2006.11477 (2020). arXiv: 2006.11477. URL: https://arxiv.org/abs/2006.11477.

[3] Alex Graves et al. "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 369–376. ISBN: 1595933832. DOI: 10.1145/1143844.1143891. URL: https://doi.org/10.1145/1143844.1143891.

[4] Awni Y. Hannun et al. "Deep Speech: Scaling up end-to-end speech recognition". In: *CoRR* abs/1412.5567 (2014). arXiv: 1412.5567. URL: http://arxiv.org/abs/1412.5567.

[5] Chengyu Sun et al. "A Review of Unsupervised Keyphrase Extraction Methods Using Within-Collection Resources". In: *Symmetry* 12.11 (2020). ISSN: 2073-8994. DOI: 10.3390/sym12111864. URL: https://www.mdpi.com/2073-8994/12/11/1864.