Projet SAM
Prédiction de tour de parole audio-langage

Eliott Crancée[1,2], Guillaume Behar[1,2] et Charlotte Aït Sadi[2]

[1] Ecole Centrale Méditerranée, 13013 Marseille, France
[2] Aix-Marseille Université, France

Lien du code source sur github `https://github.com/Znium/SAM_Project`

1 Introduction

Notre étude se concentre sur la prédiction de l'attribution de la parole dans un dialogue à deux participants. L'objectif principal est de déterminer, à la fin d'une phrase donnée, si la personne en cours d'élocution continuera de parler ou cédera la parole à son interlocuteur. À travers un modèle entraîné sur un jeu de données spécifique, nous avons observé des résultats encourageants, où le modèle a appris à anticiper le tour de parole avec une performance supérieure à un modèle initialisé au hasard. L'analyse des résultats ne suggère pas une simple corrélation aléatoire entre les entrées et les prédictions, mais plutôt une compréhension du lien entre les données d'entrée et la prédiction du tour de parole. Ces observations indiquent un apprentissage significatif du modèle, renforçant ainsi la validité de notre approche dans la résolution de la problématique de classification multimodale à deux classes.

2 Présentation du problème et des données

L'objet de la problématique réside dans la prédiction de l'attribution de la parole au cours d'une discussion. En d'autres termes, il s'agit de déterminer, en fonction des propos précédemment énoncés, à qui revient le tour de parole, c'est-à-dire qui est destiné à prendre la parole. Notre champ d'étude se restreindra à un dialogue impliquant deux participants. L'objectif consiste alors à anticiper, à la fin d'une phrase donnée, si la personne en cours d'élocution poursuivra son discours en entamant une nouvelle phrase, ou si elle cédera la parole à son interlocuteur. Nous sommes ainsi confrontés à une problématique de classification à deux classes : 0 indiquant que la personne qui s'exprime continuera de parler après la fin de sa phrase, et 1 signifiant qu'elle laissera la parole à son interlocuteur.

Les données que nous allons utiliser sont présentées de manière détaillée dans le tableau enFigure 1 ci-dessous. Les colonnes que nous allons utiliser sont : "stop" qui permet de récupérer le timecode de la fin de la phrase, "text" qui permet de récupérer ce qui vient d'être prononcé avant le timecode, "turn_after" qui indique si la personne va céder la parole ou non.

Le dataloader a été conçu de manière à générer des exemples composés de trois éléments distincts. Le premier élément consiste en un texte formé par les 20 derniers mots prononcés dans la phrase. En cas de phrase plus courte, un caractère de padding de l'encodeur textuel est ajouté pour maintenir la structure. Le deuxième élément est constitué de la dernière seconde prononcée avant la fin de la phrase dans le fichier audio. Cette caractéristique vise à représenter le ton final de la phrase, que ce soit montant ou descendant, par exemple. Enfin, le dernier élément correspond au label, à savoir la valeur de la classe, soit 0 ou 1 pour indiquer le tour de parole.

3 Présentation du modèle implémenté

Le modèle élaboré est exposé dans la Figure 2. Il se compose d'une première phase d'encodage des données, utilisant Wave2Vec2 pour les données audio et Camembert pour les données textuelles. Chacun de ces composants fait appel à un processeur dédié pour le traitement préliminaire des données avant leur encodage. Un processus de padding est appliqué entre les étapes de traitement et d'encodage afin d'assurer la cohérence du flux de données.

	ipu_id	speaker	start	stop	text	is_main_speaker	turn_at_start	turn_after	turn_start_word	yield_at_end	request_at_start	dyad
0	0	AA	4.54	4.840	tu as	True	False	True	4.84	True	False	transcr\AAOR
1	1	OR	5.14	5.825	mh ouais si tu veux	True	True	False	NaN	False	True	transcr\AAOR
2	2	OR	6.62	7.010	frog joke	True	False	False	NaN	False	False	transcr\AAOR
3	3	OR	7.42	10.870	un jour un ingénieur traversait la rue quand u...	True	False	False	NaN	False	False	transcr\AAOR
4	4	OR	11.36	13.835	si tu m'embrasses je me transforme en belle pr...	True	False	False	NaN	False	False	transcr\AAOR
...	...	...	...	...	...	...	...	...	...	...	...	...
16395	678	RPA	1117.66	1118.250	et oui c'est ça	False	False	False	NaN	False	False	transcr\RPABN
16396	679	RPA	1119.00	1123.787	bah après elles reviennent hein moi le truc de...	True	True	True	NaN	True	True	transcr\RPABN
16397	680	BN	1122.00	1122.110	ouais	False	False	False	NaN	False	False	transcr\RPABN
16398	681	BN	1122.38	1127.200	deux trois fois d'affilée je te dis c'est bon ...	True	True	False	NaN	False	True	transcr\RPABN
16399	682	RPA	1126.50	1126.790	ouais	False	False	False	NaN	False	False	transcr\RPABN

16400 rows × 12 columns

FIGURE 1 – Tableau du jeu de données utilisé.

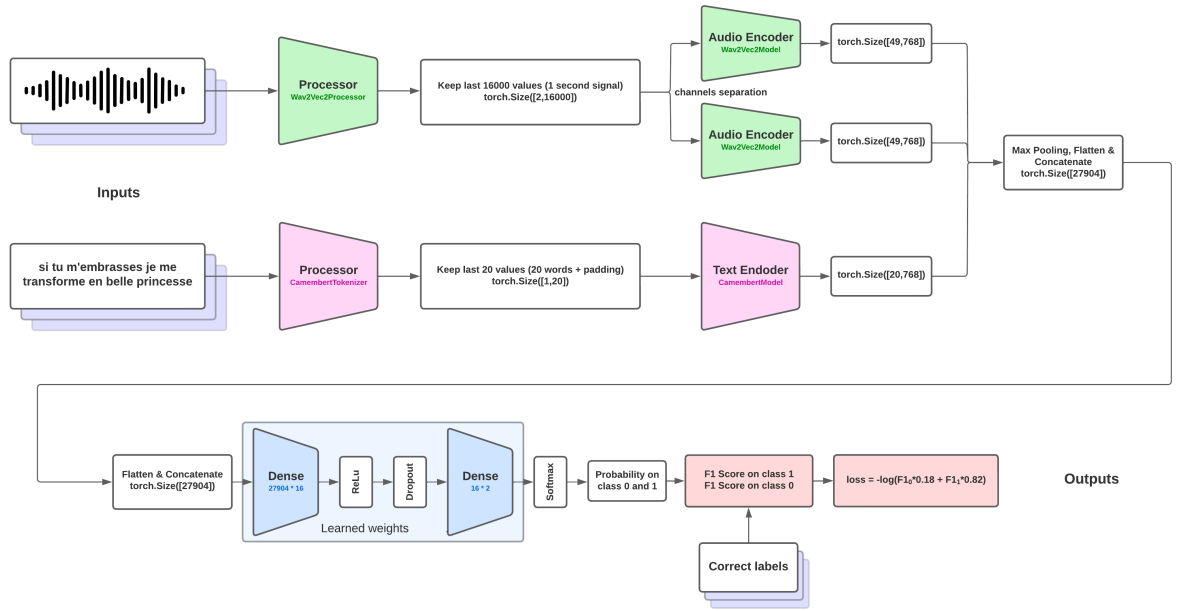

FIGURE 2 – Schéma du modèle de classification multimodale audio-langage pour le tour de parole.

Les données audio sont divisées en deux canaux distincts, attribués à chaque interlocuteur. Chacun de ces canaux est soumis à un processus d'encodage par les modèles Wave2Vec2 et Camembert. Les encodeurs génèrent des tenseurs de features, auxquels est appliqué un max pooling sur la dimension des features, particulièrement nécessaire lorsque les données audio encodées présentent une taille significative. Les tenseurs sont ensuite aplatis avant d'être concaténés en un unique tenseur unidimensionnel, conservant ainsi une taille constante.

Après l'extraction du tenseur, le modèle le transmet successivement à une couche dense de sortie de dimension 16, suivi d'une fonction d'activation ReLU. Ensuite, un dropout de 10% est appliqué avant de diriger le flux vers une deuxième couche dense de taille de sortie 2, sur laquelle est appliqué un softmax. La sortie de ce processus représente ainsi une prédiction exprimée sous forme de densité de probabilité pour les deux classes.

4 Métrique et apprentissage

Dans le cadre de la résolution de notre problème de classification, il semble judicieux d'envisager l'utilisation d'une fonction de perte de cross entropie. Les tests que nous avons effectués montrent que le modèle a tendance

à rapidement converger vers la classification d'une seule classe, indépendamment de l'exemple présenté, en l'occurrence la classe 0. En effet, dans notre jeu de données, la classe 0 représente 82% des exemples, tandis que la classe 1 ne constitue que 18%. Par conséquent, le modèle réalise que pour maximiser sa précision, la méthode la plus simple est de prédire systématiquement la classe 0, obtenant ainsi une précision de 82%, ce qui entrave la progression de l'apprentissage. Il est donc logique de considérer l'utilisation de métriques prenant en compte les faux positifs, telles que le rappel, et pour englober ces aspects, le F1 score.

$$F1 = \frac{2 \cdot (\text{précision} \cdot \text{rappel})}{\text{précision} + \text{rappel}}$$

Dans le calcul de la précision et du rappel, nous avons recouru aux valeurs de probabilités attribuées pour déterminer les vrais positifs, les faux positifs et les faux négatifs. Plus précisément, si le modèle attribue une probabilité de 0.4 à la classe 0 et, par conséquent, de 0.6 à la classe 1 pour un exemple donné, ces valeurs seront respectivement comptées comme 0.6 en vrais positifs pour le score de la classe 1. Ce même principe s'applique aux faux positifs et aux faux négatifs. Ainsi, nous calculons le score F1 pour la classification de la classe 1. Puisque l'objectif du score F1 est de converger vers 1, et qu'il est compris entre 0 et 1, nous optons pour une fonction de perte de la forme $L = -\log(F1)$ pour le transformer en problème de minimisation.

Nous avons expérimenté l'utilisation de cette fonction de perte dans le cadre de l'apprentissage, cependant, un problème inverse s'est manifesté. Le modèle a montré une tendance à prédire systématiquement la classe 1 afin de maximiser son rappel et, par conséquent, le score F1. En effet, il considère cette approche comme la méthode la plus efficace pour augmenter rapidement le score F1. Cependant, une fois cet objectif atteint, l'apprentissage stagne.

Il nous est donc venu l'idée d'utiliser un F1 score pour chacune des deux classes, et de les combiner en prenant en compte la pondération de la présence de chaque classe dans le dataset. Le calcul du F1 score ne se fait plus uniquement pour la classe 1 mais pour les deux classes. Ainsi, la nouvelle formule de la fonction de perte s'écrit :

$$L = -\log(F1_0 \times 0.18 + F1_1 \times 0.82)$$

De cette façon, le F1 score de la classe 1 est favorisé, rétablissant ainsi une équité entre les deux classes.

5 Résultats et discussion

Nous avons entrainé le modèle sur le jeu de donnée présenté en section 2. Nous ne pouvons fournir de courbe d'apprentissage, car celui-ci s'est déroulé sur plusieurs sessions, avec des tailles de dataloader variable et de batch variables. Voici donc les résultats obtenus, ici en tenant compte de la prédiction (par argmax) et non plus de la probabilité :

```
Classe 0 | Precision: 82.81%, Recall: 100.00%, F1 Score: 90.60%
Classe 1 | Precision: 0.00%, Recall: 0.00%, F1 Score: 0.00%
```

FIGURE 3 – Résultats du modèle initialisé au hasard.

```
Classe 0 | Precision: 84.26%, Recall: 58.33%, F1 Score: 68.94%
Classe 1 | Precision: 22.62%, Recall: 52.78%, F1 Score: 23.88%
```

FIGURE 4 – Résultats du modèle au début de l'apprentissage, après une petite epoch.

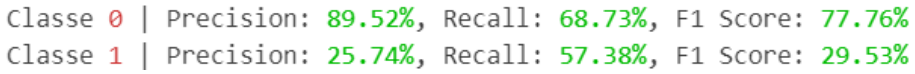

FIGURE 5 – Résultats du modèle à la fin de l'apprentissage.

On remarque que le modèle ne renvoie plus seulement une classe, mais bien une prédiction qui varie entre les deux classes. Ici, le modèle a bien appris puisqu'il fait mieux qu'un modèle initialisé au hasard. Pour autant, il est possible d'envisager que le modèle aprend simplement a renvoyer aléatoirement une classe avec une certaine probabilité pondéré pas la présence de 0 et de 1 dans le corpus d'entrainement, et donc que la sortie est décorrélé des entrées. Cependant, si c'était le cas, on aurait des résultats de precision proche de 82% pour la classe 0 et 18% pour la classe 1, ce qui n'est pas le cas. On remarque même une amélioration au cours des itérations d'epochs ce qui confirme que le modèle a bien appris un lien entre les données d'entrée et la prédiction du tour de parole.

6 Conclusion

En conclusion, notre étude sur la prédiction de l'attribution de la parole dans un dialogue à deux participants a révélé des résultats prometteurs, démontrant la capacité du modèle à anticiper le tour de parole avec une performance supérieure à un modèle initialisé au hasard. L'analyse approfondie des résultats suggère une compréhension significative du lien entre les données d'entrée et la prédiction du tour de parole.

Cependant, malgré ces avancées, des perspectives d'amélioration subsistent. Il est crucial d'explorer davantage les mécanismes internes du modèle pour garantir que l'apprentissage ne se limite pas à une simple corrélation statistique. Nous sommes confiants dans la solidité de la métrique qui a été utilisée, mais elle doit être explorée d'avantage. L'accroissement de la taille du modèle, notamment en ce qui concerne la durée d'écoute qui était auparavant limitée à une seconde, et se passer du max pooling que nous avant fait sur les features audio pourrait améliorer les performances, mais cela s'accompagne de l'ajout de beaucoup de poids dans le modèle. De plus, une période d'entraînement prolongée pourrait être envisagée.