Note méthodologique : preuve de concept

Dataset retenu

Pour évaluer nos modèles de transcription speech to text, nous avons besoin d'un dataset contenant des extraits audio et les transcriptions correspondantes. J'ai choisi le *Multilingual LibriSpeech (MLS)*, téléchargeable à l'adresse :

https://openslr.org/94

Identifier: SLR94
Summary: A large multilingual corpus derived from LibriVox audiobooks
Category: Speech
License: CC BY 4.0

Ce dataset contient des jeux de training, validation et test, pour plusieurs langues. Nos deux modèles sont préentraînés, nous allons donc utiliser une partie des fichiers test, en français.

Il existe deux niveaux de difficulté. Nous allons utiliser le plus difficile : textes littéraires, vocabulaire rare, soutenu ou archaïsant, prononciation parfois mal articulée, ... afin de tester au mieux les capacités de nos modèles.

Les concepts de l'algorithme récent

Architecture générale : Le nouvel algorithme est (comme l'ancien) un transformer encoder-decoder.

Le fichier audio est d'abord découpé en segments de 30 secondes. Les signaux sonores sont alors normalisés entre -1 et 1, et mel-encodés sur 80 canaux.

Pour chaque segment, le bloc encoder génère une représentation latente de l'input audio mel-encodé, que le decoder utilise ensuite pour produire du texte (la transcription).

Fonctionnement : Les fichiers audios sont traités au format PCM de 16 kHz, mis à l'échelle sur une plage de -1 à 1 et convertis en spectrogramme Mel à 80

canaux. La taille de la fenêtre pour la conversion en spectrogramme Mel est de 25 ms et la cadence est de 10 ms. Le spectrogramme Mel est utilisé par segments de 30 secondes chacun.

L'encodeur n'est exécuté qu'une seule fois par segment de 30 secondes pour extraire la représentation latente du spectrogramme. Tout d'abord, il applique deux fois une convolution en utilisant GELU pour l'activation avec une taille de filtre de 3 pour calculer l'encodage d'entrée. La deuxième convolution a une cadence de 2. Le transformateur effectue un encodage de position en utilisant la fonction Sin. Comme l'encodeur n'est exécuté qu'une seule fois par segment de 30 secondes, la charge n'est pas très élevée.

Le décodeur produit la probabilité d'occurrence pour chacun des 51 865 jetons (tokens) à partir de la représentation latente. Les jetons sont déterminés en effectuant une recherche gloutonne ou une recherche par faisceau sur la probabilité d'occurrence des jetons en sortie. La recherche par faisceau a une taille de faisceau (nombre de branches de recherche) de 5.

Le décodeur produit un maximum de 224 jetons par segment de 30 secondes, donc il est exécuté jusqu'à 224 fois. Si deux jetons de timestamp consécutifs apparaissent dans les résultats de décodage sur 224 inférences, la séquence de jetons de reconnaissance vocale est choisie et produite en sortie.

Pour la séquence de jetons choisie, GPT2TokenizerFast la décode en texte. L'architecture est un tokenizer de texte BPE au niveau des octets, qui ne produit pas de mots, mais des codes Unicode octets.

Le timestamp donne le nombre de secondes de reconnaissance vocale réelle qui a été effectuée. Ensuite, découper la partie non traitée du spectrogramme Mel, ajouter un spectrogramme Mel pour le porter à 30 secondes, et répéter le processus de décodage à nouveau.

Qu'est-ce qui rend ces modèles si performants ?

Le point mis en avant par les développeurs du modèle est sa capacité à travailler (être entraîné) sur un dataset beaucoup plus large que les générations de modèles précédentes.

Par exemple wav2vec 2.0, notre baseline, a été entraîné d'abord de manière non-supervisée, sur un dataset très large, pour créer son espace d'embedding, (1 000 000 d'heures de data audio). Cependant, ce modèle doit ensuite être fine-tuné de

manière supervisée, sur un dataset beaucoup plus réduit (environ 1 000 h), contenant la feature (audio) et les targets (transcripts).

Cette étape lui est indispensable, afin de créer le "dictionnaire" (la matrice d'embedding), permettant de faire le lien entre les vecteurs de l'espace d'embedding et les mots correspondants. Cette opération peut cependant diminuer la capacité de généralisation du modèle.

-> Training set, apprenants faibles
Whisper vise à atteindre des performances sans fine-tuning.

L'encodeur de Whisper a été entraîné par apprentissage faiblement supervisé, en utilisant un grand nombre de fichiers audio et de textes disponibles sur Internet. (mon hypothèse : des fichers .srt ?)

Parce qu'il est formé à partir d'un plus grand nombre d'ensembles de données que les ensembles de données académiques traditionnels, il est plus précis.
Le texte des fichiers audio disponibles sur Internet ne contient pas d'informations telles que le nombre de secondes après le début du texte, mais l'apprentissage supervisé faible permet d'apprendre sans annotation détaillée. Cela nous permet de nous entraîner sur un plus grand nombre d'ensembles de données qu'auparavant possible et d'atteindre des performances sans fine-tuning.

L'ensemble de données complet représente 680 000 heures de discours et de texte.

Source : https://medium.com/axinc-ai/whisper-speech-recognition-model-capable-of-recognizing-99-languages-5b5cf0197c16

La modélisation

L'objectif de ce projet étant d'implémenter et d'évaluer les performances d'une technologie récente par rapport à une baseline, nous n'avons pas créé ni entraîné les modèles. La tâche de transcription / traduction ne requiert pas non plus de fine-tuning supplémentaire, il n'y a donc pas de modélisation nécessaire ici, ou d'optimisation.

Feature engineering : Les modèles acceptent directement la data audio en input. Il suffit de la convertir si nécessaire, en cas de format incompatible, et de la charger avec librosa dans le cas de whisper.

L'évaluation est en revanche au cœur de notre projet, en particulier l'évaluation de leurs performances respectives en **transcription** speech to text.
La métrique standard dans l'industrie pour cette tâche est le WER, ou *Word Error Rate* (le taux de mots erronés, comportant au moins une erreur).

Cette métrique est construite à partir de la distance de Levenshtein, dont le principe est simple : la distance entre 2 chaînes de caractères correspond au nombre de transformations nécessaires pour passer de l'une à l'autre.

Voici quelques exemples pour illustrer :

```
print(_levenshtein_distance('abc', 'abcd')) # 1 transformation nécessaire, ajouter "d"
-> 1
print(_levenshtein_distance('abc', 'ABC'), '\n') # Sensible à la casse !
-> 3

print(wer('Ils sont partis', 'ils sont parti')) # taux = 2 erreurs / 3 mots (env 0.67)
-> 0.66666...
print(wer('Ils sont partis', 'i sont pati')) # Compte une erreur max par mot
-> 0.66666...
```

Le WER n'est pas une métrique "parfaite", par exemple une orthographe correcte sera sanctionnée si elle diffère du transcript de référence. Sur un dataset / corpus suffisamment important, elle permet cependant de déterminer dans quelle mesure le modèle est capable de transcrire du son en texte.

Elle constitue donc un outil de comparaison objective entre notre baseline et la nouvelle méthode.

A l'origine de ce projet, je m'intéressais surtout aux processus d'extraction de features audio, de tokenisation et d'embedding par les algorithmes de transcription récents, de type transformer. L'idée de départ était d'aider des personnes mal-entendantes à accéder à des contenus uniquement audio.

J'ai étendu le projet à un exemple d'application utile, la traduction, à la demande d'une amie restauratrice qui a parfois des problèmes de connexion internet.

Cependant la qualité des **traductions** proposées par les modèles est beaucoup plus difficile à quantifier de manière automatique, et sort du cadre initial du projet. Il existe là aussi de nombreuses métriques (on peut citer notamment BLEU, METEOR, TER, ROUGE, NIST, le WER...), qui reposent toutes sur l'utilisation de

datasets de référence. Elles ne sont donc pas utilisables sur n'importe quel jeu de données.

Notons qu'ici la transcription joue un rôle central, car sa qualité détermine la qualité possible de la traduction. En évaluant "à l'oeil" les résultats, on s'aperçoit que les conclusions semblent être en partie les mêmes, pour la transcription et pour la traduction... Mais pas tout à fait.

Une synthèse des résultats

Transcription

	model	size test set	WER moyen	WER std	time predict moyen (s)
4	whisper_large	300	0.070	0.098	74.27
1	whisper_medium	300	0.117	0.208	39.64
2	wav2vec2	300	0.139	0.115	4.05
0	whisper_small	300	0.156	0.140	12.41
3	whisper_base	300	0.263	0.165	4.62
5	whisper_tiny	300	0.383	0.181	2.8

Pour la transcription, l'analyse des résultats est simple : les modèles medium et large ont des taux d'erreur plus faibles que ceux de la baseline. Whisper large atteint des performances environ 2x meilleures que celles de wav2vec2, avec un WER moyen de 0.07 contre 0.14.

Notons cependant que ce gain en termes de performance se fait au prix d'un temps de prédiction 10 à 20x plus long.

Traduction

Il peut sembler à première vue qu'on retrouve les résultats observés ci-dessus pour la transcription seule. Voici deux exemples de (transcription +) traduction :

Texte fr d'origine : *blanqui et millière sortirent également le gouvernement n'osant pas d'abord montrer son mépris de la parole donnée le soir même du octobre avait lieu à la bourse une réunion des officiers de la garde nationale*

Baseline : *Blanqui et milliare sortirent also the government not daring to show its contempt of the parole de naissance on the evening of October had held at the bourse a meeting of officers of the National Guard*

Whisper medium : *Blanqui and Millière also went out. The government did not dare to show its contempt for the speech. On the evening of October 31, a meeting of the National Guard officers took place at the stock exchange.*

Texte fr d'origine : *il n'est pas si vilain que ça dirent les cygnes et un monsieur cygne avec un magnifique plastron blanc et de beaux pieds vernis déclara qu'il reste parmi nous et dans trois mois je lui donne ma fille en mariage*

Baseline : *It is not a villain that they say the cygnes and a mseu sign with a magnificent plastron blanc et de beaux pieds Vernis declares that he remains among us and in three months I will give him my daughter in marriage*

Whisper medium : *He's not that mean, the signs say. And a man signs with a magnificent white plaster and beautiful-glued feet, declares, that he stays among us, and in three months I will give him my daughter in marriage.*

Cependant quelques résultats surprenants sont malheureusement à relever, concernant surtout whisper large : La ponctuation disparait parfois. Plus grave : le modèle semble assez fréquemment "refuser" de traduire, se contentant de transcrire dans la langue d'origine.

En conclusion, les nouveaux modèles atteignent des **performances bien supérieures** à celles (pourtant déjà impressionnantes !) de la baseline, **pour la transcription**.

Leur supériorité au niveau de la qualité de la traduction est bien moins certaine, cependant la nouvelle méthode possède **plusieurs avantages importants** sur la baseline : notamment la détection automatique de la langue et, surtout, la gestion simple, par un seul modèle, de 57 (98) langues.

En comparaison, pour déployer une appli de traduction utilisant des modèles du type wav2vec capable de prendre en charge 57 langues, il faudrait implémenter un pipeline comprenant :

- 57 modèles de transcription, un pour chaque langue.
- 57 x 56 (= 3 192) modèles traduction...

Cette solution est tellement plus lourde à maintenir en production (et scale) que google (dont le modèle seq2seq est assez similaire à wav2vec) a suspendu en 2011 l'accès à l'API de translate, la jugeant trop coûteuse, et préférant se concentrer sur l'appli mobile.

L’analyse de la feature importance

Les modèles utilisés ici prennent une seule feature en input, une analyse traditionnelle de la feature importance n’aurait donc pas beaucoup de sens ici. Nous reviendrons cependant en conclusion sur les méthodes utilisées pour accéder aux représentations internes des réseaux neuronaux en général (problème de la “boîte noire”) et des transformers en particulier.

Les limites et les améliorations possibles

Pour gagner en performance, je vois 4 points principaux à prendre en considération. On peut en effet souhaiter :

- améliorer les performances générales du modèle (meilleure transcription),
- réduire les temps de prédiction,
- prendre en charge davantage de langues,
- Ajouter des fonctionnalités ou capacité supplémentaires, comme la translitération (changement de langue au cours d’une phrase par exemple).

Difficile d'imaginer quelles inovations, ou quelles applications nouvelles ou améliorations de technologies existantes vont pouvoir révolutionner le domaine, demain ou dans un an.

Voici malgré tout une idée de piste à explorer, l'avenir dira si elle se révèle utile.

L'extraction de feature par mel-encoding

Je suis un peu étonné que le traitement des données audio par whisper repose sur un mel frequency encoding au préalable, qui lui sert de preprocessing function. Je soupçonne que cela lui permet d'utiliser un dataset plus grand, et que cela n'a pas bcp d'importance ds ts les cas, le mel spectrogramme étant une méthode d’encoding très efficace, et un standard très largement utilisé dans l'industrie.

On peut cependant rappeler 2 choses :

1) le mel encoding, un peu comme la méthode sift dans le domaine des images, est un processus d'extraction de features audio élaboré de manière en partie empirique et en partie arbitraire.

Quelle que soit la qualité des features obtenues, il y a fort à parier que, (exactement comme pour les modèles de classification d'image / vision par exemple) pour les prochaines générations, toute l'extraction de feature sera faite par le modèle lui-même, avec des premières couches / modules de data augmentation + extraction intégrée.

Cette méthode, déjà largement implémentée dans le domaine de la compréhension d'images, présente de nombreux avantages (performance, compatibilité, simplicité) et me semble très prometteuse ici aussi, d'autant plus que :

2) L'application idéale du mel encoding, son objectif premier, n'est pas exactement d'opérer une extraction de features optimale pour un modèle de machine learning. Pas du tout en fait, cette technique ayant été inventée à la fin des années 1930.

Il s'agit plutôt de traiter un signal sonore destiné à être entendu par une oreille humaine. Cela en fait un très bon "filtre" pour l'industrie musicale par exemple, puisqu'il est conçu, d'une part pour éliminer (atténuer) tous les sons inaudibles pour l'oreille humaine (trop grave, trop aigu, trop peu de volume), ce qui en fait un excellent outil de compression, tout en conservant d'autre part les sons que nous entendons.

En ce qui concerne **l'interprétabilité** des prédictions de ce type de modèles, il s'agit d'un champ de recherche en plein développement. Une approche prometteuse, appliquée récemment au problème de l'évaluation des capacités des LLMs à jouer aux échecs, consiste à implanter au réseau neuronal, en parallèle de son système d'auto-attention, un réseau de sondes conçu pour nous renseigner sur les **représentations internes** du modèle.