> *Compte rendu de la méthode suivie jusqu'à présent ainsi que des difficultés rencontrées.*

Travail sur le Corpus du Français Parlé Parisien (CFPP)

Les transcriptions du CFPP ayant été réalisées par différents transcripteurs selon différentes règles, le corpus nécessitait d'être uniformisé et les transcriptions nettoyées : pour cette étape, j'ai opéré simplement avec des expressions régulières me permettant notamment de supprimer les indications des transcripteurs qui ne nous intéressent pas, ou bien par exemple de rajouter les clitiques manquants. (Pour plus de détails sur les transcriptions, voir le fichier `annexe.md` sur le git.)

Après avoir testé un entretien selectionné au hasard (locutrice : Anita Musso) sur deux des aligneurs automatiques qui me semblaient les plus pertinents (WebMaus et MFA), c'est finalement MFA (Montreal Forced Aligner) que je retiens, tant pour la qualité de ses résultats que pour ses fonctionnalités diverses (modification du dictionnaire...). Cependant, afin d'optimiser les résultats obtenus, il faudrait passer à l'aligneur des fichiers correspondant à des intervalles de parole de moins de 10s (5s serait l'idéal, 15s le maximum possible ici). Néanmoins, pour de nombreux entretiens, les intervalles de parole segmentés par les transcripteurs peuvent être très longs (dans le pire des cas : plus d'une minute, voire même une minute trente) : la segmentation en intervalles de paroles est donc très hétérogène à travers le corpus, et certaines fois même à travers les fichiers de transcription .TextGrid. Ainsi, il faut re-segmenter les intervales de parole "longs", en intervalles courts, sur praat. Il s'agit là de la partie la plus chronophage du travail : pour les fichiers les "mieux" segmentés, il suffit de deux à trois heures pour passer les transcriptions en revue, et effectuer une nouvelle segmentation, mais dans le pire des cas (ex : Yvette Audin), on peut compter sur 7h et plus de travail. Le temps passé sur cette étape dépend également de la longueur de l'entretien.

À partir des TextGrids segmentés et nettoyés (on conserve uniquement la tier du locuteur sujet), on génère des corpus (un par locuteur) consitués de fichiers .txt et .wav (une paire de fichier par intervale de parole donc). J'avais dans un premier temps choisi de générer des fichiers .TextGrid, mais j'ai finalement rencontré plusieurs problèmes avec MFA :

- impossibilité de valider le corpus d'entrée (MFA décrète qu'il ne reconnaît pas de fichiers audio dans le corpus, mais étrangement cela serait dû à un problème venant des fichiers .TextGrid)
- impossibilité de réaliser l'alignement automatique (pour les mêmes raisons que plus haut)
  Finalement le problème viendrait donc de la façon dont les fichiers .TextGrid sont

générés : dans mon cas, j'ai notamment testé deux modules python différents, `textgridtools` et `praatio`. Les deux modules gèrent les fichiers TextGrid différemment, avec différentes classes, proposant à l'utilisateur différentes structures afin de modifier ou créer le fichier TextGrid désiré. Après de nombreuses expérimentations avec les deux modules, il me semblait que `textgridtools` était plus pratique, tandis que le travail avec `praatio` devenait redondant. Cependant, il semblait que c'était justement les .TextGrid générées avec `textgridtools` qui posaient problème à MFA, pour des raisons qui me sont un certain temps restées inconnues. (À un moment donné, il semblait que le problème venait de la "structure" des .TextGrid. Pour un "intervale de parole", on devrait avoir en fait trois intervales ; il fallait donc coder cette structure afin d'obtenir le fichier désiré. Mais même après avoir appliqué cela, MFA refusait de lire les fichiers.)
Par besoin de simplicité et de gain de temps, j'ai donc finalement opté pour la génération des fichiers .txt simples. Le nom des fichiers contient finalement toutes les informations nécessaires (nom du locuteur, start_time, end_time). Par ailleurs, les fichiers obtenus sont moins volumineux, et MFA semblent les traiter plus rapidement (pour la validation du corpus du moins).

Il est évident qu'il reste encore beaucoup de fichiers à traiter : j'aurai aimé pouvoir aller plus vite, mais je pense à présent avoir trouvé la bonne méthode pour traiter les fichiers à venir d'autant plus rapidement.

Analyse avec openSMILE

Pour l'instant, j'ai uniquement travaillé avec le module python `opensmile` : de cette façon, je peux dirrectement procéder à l'analyse des données obtenues (*extracted features*) avec `pandas`. La prise en main de l'outil m'a pris un petit peu de temps car j'ai eu besoin de me refamiliariser avec certaines manipulation avec `pandas`, mais cela m'a permis de revoir un grand nombre de notions qui me seront utiles pour la suite. De la même façon, j'ai pris le temps d'apprendre à utiliser `seaborn`, afin d'anticiper les diverses représentations graphiques que je pourrais également réaliser.
Ainsi, j'ai pu réaliser des premiers graphiques relativement simple des formants (F1, F2, F3) à partir de données extraites d'openSMILE sur un segment d'audio choisi.
J'ai passé également un peu de temps à réfléchir comment extraire un phonème donné à travers tout le corpus d'un locuteur, pour ensuite en extraire des mesures avec openSMILE, pour finalement les traiter. J'ai donc une plutôt bonne idée du cheminement à suivre pour le moment.