Développement de technologies vocales pour quelques langues africaines

Rapport de stage

Nom : GAUTHIER
Prénom : Elodie

UFR SHS - IMSS

Master 2 Professionnel

Spécialité : Ingénierie de la Cognition, de la Création et des Apprentissages

Parcours : Double Compétence en Informatique et Sciences sociales

Sous la direction de Laurent Besacier

Année universitaire 2013-2014

Table des matières

Introduction

D'après l'UNESCO [Unesco, 2010], « le nombre de langues parlées en Afrique va de 1 000 à 2 500, selon les estimations et les définitions. Les États monolingues n'existent pas et les langues traversent les frontières sous forme de configurations et de combinaisons différentes. Le nombre de langues varie entre deux et trois au Burundi et au Rwanda, à plus de 400 au Nigeria. ». C'est dans ce contexte que se situe le développement de technologies vocales sur mobiles pour les langues africaines : les systèmes de reconnaissance vocale et de synthèse vocale peuvent aider les personnes analphabètes à communiquer mais peuvent aussi être utilisées pour des applications ludiques. De plus, l'adjonction d'un système de traduction automatique peut faciliter les échanges commerciaux.

Ce stage fait suite au projet professionnel réalisé dans le cadre du projet ANR ALFFA, qui a pour but de développer des technologies vocales pour téléphones mobiles en Afrique. Durant ce stage, nous avons effectué de nouveaux tests afin de voir comment peut être amélioré notre précédent système automatique de reconnaissance d'une grande langue véhiculaire d'Afrique de l'Ouest : le hausa. Pour créer nos nouveaux modèles de comparaison, nous avons continué à utiliser la boîte à outils de reconnaissance de la parole Kaldi.

Ce rapport est découpé en 6 chapitres. Le premier chapitre présente le contexte du projet ; le deuxième rappelle les principes de fonctionnement d'un système de reconnaissance automatique de la parole ainsi que le corpus de travail exploité; le troisième explique les nouvelles expérimentations menées ainsi que leurs résultats; le quatrième présente la plateforme GitHub sur laquelle nous avons partagé nos travaux; le cinquième fait le bilan sur les techniques apprises et acquises au cours de ces cinq mois de stage long mais est aussi un bilan personnel sur le travail effectué.
Le planning de travail est visible en annexe 1.

Chapitre 1 – Contexte du projet

Ce projet professionnel a été réalisé au sein du Laboratoire d'Informatique de Grenoble et plus précisément au sein de l'équipe GETALP.

1.1. Le Laboratoire d'Informatique de Grenoble

Le Laboratoire d'Informatique de Grenoble (LIG) se situe sur le campus universitaire de Saint-Martin d'Hères. Il est axé sur cinq thématiques de recherche :

- Génie des Logiciels et des Systèmes d'Information ;

- Méthodes Formelles, Modèles et Langages ;

- Systèmes Interactifs et Cognitifs ;

- Systèmes Répartis, Calcul Parallèle et Réseaux ;

- Traitement de Données et de Connaissances à Grande Echelle.

Les défis du LIG – communs à ces cinq axes de recherche – sont : la diversité et la dynamicité des données, des services, des dispositifs d'interaction et des contextes d'usage imposent l'évolution des systèmes et des logiciels pour en garantir des propriétés essentielles telles que leur fiabilité, performance, autonomie et adaptabilité.

1.2. L'équipe GETALP

Le GETALP[1] (Groupe d'Étude en Traduction Automatique/Traitement Automatisé des Langues et de la Parole) est une équipe parmi les 22 présentes du Laboratoire d'informatique de Grenoble. Elle rassemble des informaticiens, linguistes, phonéticiens, traducteurs et traiteurs de signaux.

Les travaux de recherche du GETALP sont pluridisciplinaires et participent à la création d'une informatique « ubilingue ». C'est pourquoi les domaines de recherche sont variés et font partie aussi bien du domaine de l'informatique, que des sciences du langage.

[1] http://getalp.imag.fr/

Parmi les 9 axes de recherche de l'équipe, mon travail s'insère dans celui de la « Traduction et transcription automatiques de la parole ».

1.2.1 Le projet ALFFA

Le projet ALFFA (*African Languages in the Field: speech Fundamentals and Automation*) est un projet ANR, financé sur 4 ans, qui a débuté en octobre 2013. Ce projet vise à proposer, à terme, des micro services vocaux pour les téléphones mobiles en Afrique (par exemple, un service de téléphone pour consulter le prix des matières premières ou pour fournir des rapports vocaux des systèmes d'information) mais également des outils de traitement automatique de la parole afin d'aider les linguistes de terrain à décrire les langues et à les analyser.
4 organismes participent à ce projet : le Laboratoire d'Informatique de Grenoble (i.e. : LIG), le Laboratoire d'Informatique d'Avignon (i.e. : LIA), le laboratoire Dynamique du langage à Lyon (i.e. : DDL) et Voxygen une jeune entreprise bretonne spécialisée en synthèse vocale. C'est donc un projet interdisciplinaire puisqu'il réunit non seulement des informaticiens et spécialistes en traitement du signal mais également des linguistes de terrain et des phonéticiens.

1.3 Le Hausa

Le hausa fait partie de la famille des langues chamito-sémitiques. Plus précisément, le hausa est la langue la plus parlée des langues tchadiques [Vycichl, 1990]. Il est la langue officielle du nord Nigeria (30 millions de locuteurs environ) et est une langue nationale du Niger (9 millions de locuteurs environ) mais est aussi parlé au Ghana, au Benin, au Cameroun, au Togo, au Tchad et au Burkina Faso [Koslow, 1995]. Le hausa est considéré comme une langue véhiculaire d'Afrique de l'Ouest et d'Afrique centrale : il est parlé dans de multiples grandes villes commerciales telles que Dakar, Abidjan, Lomé, Ouagadougou ou encore Bamako.

Environ un quart des mots du hausa provient de l'arabe mais la langue a aussi été influencée par le français. Le hausa peut être écrit avec l'orthographe arabe depuis le début du 17è siècle : ce système d'écriture est appelé *'ajami.* Cependant, l'orthographe officielle est basée sur l'alphabet latin et est appelé *boko.* Ce système d'écriture a été imposé par les anglais lors de la colonisation, dans les années 30.

Le *boko* est formé de 22 caractères provenant de l'alphabet anglais (i.e. : A/a, B/b, C/c, D/d, E/e, F/f, G/g, H/h, I/i, J/j, K/k, L/l, M/m, N/n, O/o, R/r, S/s, T/t, U/u, W/w, Y/y, Z/z) plus ɓ, ɗ, ƙ (respectivement transcrits par B/b, D/d, et K/k dans les journaux) et ' (qui représente l'arrêt glottal).

A l'oral, il existe 3 tons différents en Hausa : l'intonation de la voix est distinctive d'une syllabe à une autre. Ces deux particularités de la langue portent sur les voyelles. Chacune des 5 voyelles /a/, /e/, /i/, /o/ et /u/ peuvent avoir un ton bas, haut ou descendant. Mais aussi, les voyelles peuvent varier en durée (i.e. : voyelle longue ou courte) et cette variation à une incidence sur la signification du mot. Concernant l'écrit, les tons et longueurs des voyelles sont marqués en *'ajami*, mais pas en *boko*. Le *boko* étant l'écriture standard, ceci pose problème lors de la construction de dictionnaire – au niveau de la transcription phonétique – car les transcriptions diffèrent. Lorsqu'ils sont représentés, ces tons et durées sont décrits par des diacritiques (i.e. : l'accent grave pour le ton bas, l'accent aigu pour le ton haut et l'accent circonflexe pour le ton descendant, le doublement de la voyelle si elle est longue) dans les dictionnaires.

Chapitre 2 – Rappels sur la reconnaissance automatique de la parole

2.1. Principes de la reconnaissance automatique de la parole

La reconnaissance automatique de la parole (i.e. : « RAP ») consiste à transcrire un signal acoustique en une suite de mots écrits. Par la suite, le système est capable d'analyser et d'interpréter ces séquences afin de prendre une décision face à ce qu'il a reconnu.

La construction d'un système de reconnaissance automatique de la parole peut être décomposée en deux parties: l'extraction des paramètres acoustiques – les MFCCs (i.e. : *Mel-Frequency Cepstrum Coefficients* ou coefficients Mel cepstraux) – et l'étape de modélisation. Cette dernière comprend deux étapes : le modèle acoustique et le modèle de langage, tous deux créés à partir de HMMs (i.e. : *Hidden Markov Model* ou chaînes de Markov cachées).

Ci-dessous, nous pouvons voir un schéma du processus de reconnaissance :

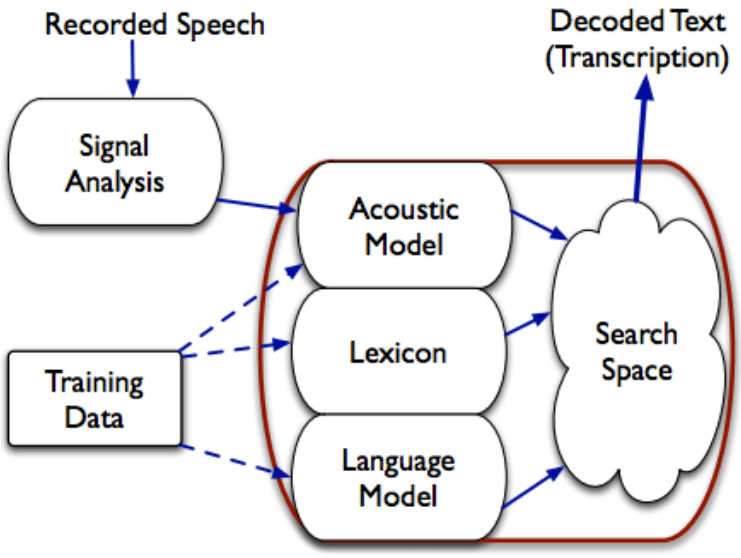


Figure 1 Schéma du processus de reconnaissance d'un système de reconnaissance automatique de la parole

2.1.1 La paramétrisation (extraction des paramètres acoustiques)

La première étape d'un système de reconnaissance de la parole est d'extraire des caractéristiques qui permettent de discerner les composants du signal audio qui sont pertinents pour l'identification du contenu linguistique, en rejetant les autres informations contenues dans ce signal (i.e. : le bruit ambiant, les émotions, etc.).
Les sons émis par un être humain sont représentés par la forme du tractus vocal, incluant la langue, les dents, les lèvres, etc. Par conséquent, en déterminant précisément cette forme, il est possible d'identifier le phonème produit.
Les MFCCs permettent de représenter avec justesse l'enveloppe du spectre de puissance à court-terme. Ce sont ces coefficients que nous allons donc extraire, puisque la forme du conduit vocal se manifeste dans cette enveloppe. Ainsi, le système de reconnaissance ne se sert pas directement du signal sonore qu'il reçoit en entrée : il exploite la représentation paramétrique de ce signal, c'est-à-dire que le signal acoustique est découpé en vecteurs de paramètres acoustiques (i.e. : les coefficients Mel cepstraux).
Cette étape d'extraction permet de représenter digitalement des signaux d'entrée et de faciliter le traitement des étapes d'apprentissage et de décodage. Plus le nombre de paramètres est important, plus l'étape d'apprentissage sera performante.

2.1.2 L'identification (construction des modèles)

2.1.2.1 Le modèle acoustique

Le modèle acoustique, représenté par des chaînes de Markov cachées, provient d'un entraînement réalisé à partir d'enregistrements de corpus oraux.

Selon les caractéristiques du locuteur – accent, âge, condition physique, sexe, etc. –, la prononciation des phonèmes est différente. Afin de modéliser ce phénomène, ce sont en général des HMMs à 3 états qui sont utilisés. À chaque état correspond un modèle de mélange gaussien (i.e. : GMM) qui permet, lors de l'étape de décodage, d'émettre une hypothèse sur la réalisation du phonème. Dans le cas des modèles acoustiques en contexte, chaque phonème possède un modèle par contexte existant.

Après l'étape d'apprentissage, le système décode acoustiquement de nouveaux enregistrements : le modèle acoustique attribue, sur les nouvelles données, des scores

de vraisemblance « donnée apprise – donnée inconnue » s'appuyant sur les modèles HMMs appris. Les scores permettent d'extraire des mots grâce à un dictionnaire de prononciation. Ce dictionnaire contient les mots présents dans le corpus oral utilisé (ou plus) suivis de leur prononciation phonétique ainsi que toutes les réalisations acoustiques possibles d'un phonème en fonction du contexte (allophones).

2.1.2.2 Le dictionnaire de prononciation

Le dictionnaire acoustique joue un rôle essentiel dans la reconnaissance automatique de la parole. En effet, c'est lui qui permet de faire le lien entre la représentation acoustique d'un mot et sa représentation phonémique.
Le dictionnaire est utilisé par deux fois lors du processus de reconnaissance. Une première fois, durant l'étape d'apprentissage : il permet de construire des modèles acoustiques en faisant le lien entre les représentations acoustiques de chaque unité lexicale et leurs représentations phonémiques. Une deuxième fois, lors de l'étape de décodage : il permet au système de reconnaître les unités acoustiques, de leur attribuer une unité phonémique, afin de les assembler pour former un mot.

2.1.2.3 Le modèle de langage

Les suites de mots obtenues lors du décodage sont analysées avec un modèle de langage. Ce modèle est réalisé à partir des transcriptions du corpus oral. Il permet d'estimer la probabilité qu'une séquence de mots existe dans la réalité et dépend directement de la langue à reconnaître ainsi que de la grammaire de cette langue. Afin d'être généré, le modèle de langage s'appuie sur le dictionnaire de prononciation précédemment cité.

2.2. La boîte à outils Kaldi

Kaldi est une boîte à outils open source pour la reconnaissance de la parole, qui permet de construire un système de reconnaissance. Il est écrit en C++, sous licence

Apache v2.0. Kaldi se veut simple d'utilisation, le plus générique et le plus flexible possible pour que l'utilisateur puisse personnaliser son code comme il le désire.

2.3. Le corpus de travail

Concernant les données audio, nous avons exploité celles enregistrées par Tim Schlippe [Schlippe, 2012]. Ces données ont initialement été collectées dans le but de développer un système de reconnaissance de la parole continue à grand vocabulaire.
Tim Schlippe et son équipe ont extrait des phrases depuis différents sites web en Hausa et les ont mises en forme dans le but d'être exploitables pour le modèle de langue. Puis, il les ont faites lire à 103 locuteurs natifs du Cameroun (i.e. : 33 hommes et 69 femmes de 16 à 60 ans). Les données ont été collectées selon le même schéma que dans GlobalPhone[2]. Au total, ce sont 7 895 expressions qui ont été enregistrées. Ces enregistrements ont été réalisés avec un micro-casque Sennheiser 440-6, dans un environnement sans bruit. Les données ont été échantillonnées à 16kHz, en 16-bit et encodées au format PCM.
Ci-contre, le tableau détaillant le découpage du corpus oral transcrit pour le Hausa.

Ensemble	Hommes	Femmes	#occurrences	#tokens	dureé
Apprentissage	24	58	5863	40k	6 h 36 m
Développement	4	6	1021	6k	1 h 02 m
Evaluation	5	5	1011	6k	1 h 06 m
Total	33	69	7895	52k	8 h 44 m

Figure 4 : Détail du corpus oral transcrit pour le Hausa

Pour ce qui est du dictionnaire de prononciation, nous avons utilisé celui créé par Globalphone. Il contient 42 662 entrées dont 42 079 mots (des mots peuvent avoir plusieurs prononciations). Il répertorie les 33 phonèmes de la langue (i.e. : 26 consonnes, 5 voyelles et 2 diphtongues).
Le Hausa est une langue à tons et les voyelles peuvent varier en durée, selon la structure de la syllabe.

[2] GlobalPhone est un corpus multilingue de données orales et écrites, disponible pour 20 langues depuis ELRA (http://catalog.elra.info)

2.4. Rappel sur le taux d'erreurs mots (WER)

Le taux d'erreurs mots (i.e. : *Word Error Rate*) sert à évaluer le taux de reconnaissance du système après décodage. Cette mesure permet d'attribuer un poids à chaque opération effectuée par le système lorsque il transforme la référence (i.e. : la transcription du signal audio qu'il faut reconnaître) en l'hypothèse (i.e : la solution la plus probable trouvée par le système). Il existe trois sortes d'opérations : la substitution, la suppression et l'insertion. Ainsi, le système compte le nombre d'opérations effectuées sur l'alphabet (ici, chaque élément de l'alphabet est un mot) et attribue un score pour chacune des séquences de phrases analysées.

A noter que nous pouvons aussi calculer le taux d'erreurs de caractères (i.e. : *Character Error Rate)* grâce à cette même méthode (dans ce cas, chaque élément de l'alphabet sera un caractère).

Pour obtenir le WER ou le CER d'un système, nous avons utilisé le programme Sclite[3] créé par l'institut NIST (National Institute of Standards and Technology). Cet outil s'utilise en ligne de commande ou dans un script *shell* ; il permet de donner une note à un système de reconnaissance automatique de la parole en comparant la sortie du décodage (appelée hypothèse) avec sa référence textuelle (i.e. : la transcription orthographique associée à l'enregistrement décodé). Cet outil permet donc d'évaluer les performances d'un système de reconnaissance automatique de la parole. Nous avons donc modifié un script précédemment utilisé par l'équipe afin de l'adapter à nos propres données. Un exemple de script est visible en annexe 2.

De plus, pour rechercher et lister les erreurs sur les mots, nous avons utilisé les commandes `cat` et `sed`. Cela nous a permis de ne garder que ce que nous voulions : les lignes qui correspondent à la référence dans le fichier de sortie du décodage. La ligne ci-dessous en est un exemple:

```
cat recap_sclite_result_align.txt | sed '/Eval/d' | sed '/Scores/d' | sed '/Speaker/d' | sed '/System/d' | sed '/DUMP/d' > toto
```

[3] http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sclite.htm

Les mots erronés sont codés en majuscules. Nous avons alors écrit un script python qui permet de supprimer les mots corrects (en minuscule) pour ne garder qu'eux. Ce script est visible en annexe 3. Nous avons ensuite pris la sortie de ce script et nous avons appliqué les commandes `uniq` et `sort` afin de les classer selon le nombre d'occurrences. Au moyen d'un autre script python que nous avons écrit, nous avons aussi pu calculer le nombre d'occurrences total de chaque mot de la référence. Ceci nous a permis d'évaluer les mots les plus mal reconnus en fonction de leur fréquence et de vérifier s'il existe une corrélation entre les erreurs et les tons ou longueurs que comportent un mot.

Chapitre 3 – Les expérimentations

3.1. Taux d'erreurs mots des différents systèmes selon la prise en compte ou non des tons et longueurs de voyelles

Tout comme a procédé Tim Schlippe et son équipe, nous avons créé différents modèles afin d'évaluer l'impact des longueurs des voyelles et des tons présents dans la langue Hausa sur le système de reconnaissance. Pour réaliser ceci, nous avons modifié le dictionnaire de prononciation que nous exploitons depuis le début (i.e. : le dictionnaire tiré du corpus GlobalPhone) afin qu'il inclut ou non les tons et longueurs des voyelles.

Nous pouvons observer les différents résultats obtenus d'après le tableau ci-contre. Les résultats correspondent tous à des systèmes créés à partir du modèle de langage tiré du corpus GlobalPhone convertit en lettres minuscules (pour ne pas faire de distinction sur la casse) et du modèle acoustique le plus performant (i.e. : celui à base de SGMM+MMI).

Dictionary	WER (%) on dev
system3 *(no tones, no vowel length)*	9.52
system8 *(no tones, vowel length)*	9.53
system7 *(tones, no vowel length)*	10.30
system5 *(tones and vowel length)*	10.20

Tableau 1 - WER des systèmes en fonction de la prise en compte des tons et longueurs des voyelles

Ainsi, le modèle qui obtient le meilleur score est celui dont le dictionnaire ne prend en compte aucune marque de ton ou de longueur, avec 9.52% d'erreurs seulement. Aussi, le modèle dont le dictionnaire n'inclut que les marques de longueurs obtient un score de 9.53%, ce qui est très satisfaisant. A l'inverse, le plus mauvais est celui qui n'intègre que les tons et qui exclut les longueurs dans le dictionnaire de prononciation.

Même si les scores restent proches, nous nous rendons compte que les tons posent problème au système. En revanche, le système n'a pas plus de difficultés à traiter la longueur des voyelles.

3.2 Modification du dictionnaire acoustique

En analysant les scores obtenus pour le system5 (i.e. : dictionnaire incluant les tons et les longueurs de voyelles), nous nous sommes étonnés de la performance moins bonne du modèle par rapport à son opposé (i.e. : celui qui utilisait le dictionnaire de prononciation excluant les tons et longueurs de voyelles.). En regardant plus en détails le dictionnaire, nous avons remarqué que certaines voyelles étaient courtes, d'autres longues, d'autres neutres, d'autres portant des tons. Cependant, après plusieurs recherches sur la structure phonologique de la langue Hausa, nous avons trouvé que les tons et longueurs pouvaient porter sur une voyelle. En effet, une voyelle pouvait être étiquetée non seulement par un ton mais également par une longueur, alors que dans le dictionnaire récupéré de GlobalPhone les étiquettes concernant les tons et longueurs de voyelles étaient mutuellement exclusifs. Aussi, en Hausa, une voyelle n'a que deux états (elle est soit brève soit longue) alors que dans le dictionnaire en notre possession une voyelle peut en avoir trois (bref, neutre, long).

Nous avons alors modifié le dictionnaire afin qu'il représente mieux la réalité de la langue. Nous avons supprimé l'étiquette « S » qui permettait de reconnaître une voyelle courte. Ainsi, les voyelles qui n'avaient pas d'étiquette (i.e. : les voyelles que nous avons considérées « neutres ») sont amalgamées avec les voyelles auparavant étiquetées « S » *(short)* afin qu'il n'y ait plus qu'une catégorie de longueur de voyelle étiquetée (i.e. : les voyelles longues). Ainsi, le système devrait avoir un choix plus juste lors du décodage pour les mots contenant des voyelles.

Finalement, comme nous pouvons le voir dans le tableau ci-dessous, les performances du système ne sont pas meilleures. Les scores sont très proches mais l'élimination de l'étiquette distinguant une voyelle courte n'est pas plus pertinente en terme de performance. Néanmoins, le dictionnaire modifié reflète mieux la langue Hausa d'un point de vue phonologique.

Dictionary	WER (%) on dev
system5 *(baseline)*	10.20
system10 *(modified)*	10.60

Légende des différents systèmes :

System5 : longueur et tons (dictionnaire de base)

System10 : longueur et tons (dictionnaire amélioré)

3.3. Evaluation des marques de longueur dans le dictionnaire acoustique

En comparant les performances des systèmes selon la prise en compte des tons et longueurs dans le dictionnaire acoustique, nous nous sommes aperçus que le modèle basé sur le dictionnaire qui ne comprenait pas les marques de tons mais uniquement les marques de longueurs était celui qui obtenait le meilleur score. Puisque ce modèle avait été fondé sur le dictionnaire initial, nous avons voulu tester si le modèle basé sur notre dictionnaire de prononciation modifié obtenait un meilleur score.

Le system8 a été construit à partir du dictionnaire acoustique de base, c'est-à-dire celui qui inclut des voyelles « neutres » (sans information sur leur longueur (e.g. : a)), des voyelles courtes (avec une étiquette notée « *S* » (e.g. : a**S**)) ainsi que des voyelles longues (avec une étiquette notée *« L »* (e.g. : a**L**)).
Le system8bis a été construit à partir du dictionnaire de base, épuré de l'étiquette signifiant qu'une voyelle est courte. En effet, comme stipulé en 3.4, la marque notant une voyelle courte nous a paru redondante.

Le tableau ci-dessous compare les résultats des deux systèmes :

	WER (%) on dev
system8 *(1st version of the dictionary)*	9.53
System8bis *(dictionary modified)*	9.49

Nous pouvons observer que le système basé sur le dictionnaire modifié obtient un meilleur score que celui construit à partir du dictionnaire de base. Lorsque nous excluons les tons, nous arrivons à avoir un bon système de reconnaissance.

Les résultats obtenus en 3.1 et en 3.2, adjoints à ceux-ci, permettent de confirmer notre première conclusion sur la difficulté du système à reconnaître les tons en Hausa.

3.4. Modification de la valeur du poids des insertions lors de l'étape de décodage

Nous nous sommes rendus compte lors de l'analyse des hypothèses faites par le système au moment du décodage que les erreurs étaient souvent commises sur des mots qui pourraient être formés par deux mots courts du dictionnaire.

Nous avons modifié la variable `word_ins_penalty` du script `score.sh` du toolkit Kaldi qui permet de modifier le poids accordé aux insertions lors du décodage pour voir si cela influait sur les performances et de quelle manière.

Le tableau ci-contre montre les résultats obtenus pour chaque modèle de système créé. Tout comme les résultats présentés précédemment, les scores correspondent au système créé à partir du modèle de langue de GlobalPhone sans la casse et du modèle acoustique à base de SGMM+MMI.

	WER (%) on dev
system3 *(word_ins_penalty = 0)*	9.52
system6 *(word_ins_penalty = 3.0)*	11.93
system9 *(word_ins_penalty = 1.5)*	10.49

Tableau 2 - WER des différents systèmes selon le poids accordé aux insertions lors du décodage

Définition des différents systèmes :

System3 : pas de longueur, pas de tons + poids de décodage = 0.0

System6 : pas de longueur, pas de tons + poids de décodage = 3.0

System9 : pas de longueur, pas de tons + poids de décodage = 1.5

D’après le tableau ci-dessus, nous pouvons voir que les scores ne sont pas meilleurs. Toutefois, certaines références ont été mieux décodées, comme le montre les exemples ci-dessous.

	system3	system6	system9
REFERENCE	itama ingila bata kayatar da 'yan kallo ba		
HYPOTHESE	ITA MA ingila BA TA KAYATA da 'yan kallo ba	itama ingila bata kayatar da 'yan kallo ba	itama ingila bata kayatar da 'yan kallo ba
REFERENCE	saboda babu hanyoyi masu kyau		
HYPOTHESE	SABO DA babu hanyoyi masu kyau	saboda babu hanyoyi masu kyau	SABO DA babu hanyoyi masu kyau

Tableau 3 - Exemples de sorties de décodage mieux reconnues avec un poids de 3

En contrepartie, les mots courts sont désormais mal reconnus et remplacés par des mots plus longs puisque le taux d’insertion de mots à été réduit. Ci-contre, nous avons donné deux exemples :

	system3	system6	system9
REFERENCE	itama ingila bata kayatar da 'yan kallo ba		
HYPOTHESE	ITA MA ingila BA TA KAYATA da 'yan kallo ba	itama ingila bata kayatar da 'yan kallo ba	itama ingila bata kayatar da 'yan kallo ba
REFERENCE	saboda babu hanyoyi masu kyau		
HYPOTHESE	SABO DA babu hanyoyi masu kyau	saboda babu hanyoyi masu kyau	SABO DA babu hanyoyi masu kyau

Tableau 4 - Exemples de sorties de décodage moins bien reconnues avec un poids de 3

Finalement, la modification du poids d’insertions n’améliore pas les performances du système. Un poids laissé à 0.0 reste un bon compromis.

3.5. Evaluation du taux d'erreurs de caractères entre le système sans tags versus le système avec tags

Afin de calculer le taux d'erreurs de caractères, nous avons de nouveau utilisé le programme Sclite, cité précédemment en 2.4. Nous l'avons modifié afin qu'il ne prenne plus en compte les mots mais les caractères des sorties de décodage.

Pour calculer la fréquence des caractères, nous avons procédé de la même façon que pour les mots, à la différence que l'on a isolé les caractères erronés et non plus les mots.

Nous avons comparé le taux d'erreurs de caractères entre le modèle généré à partir du dictionnaire de prononciation excluant les tags (i.e. : system3) et le modèle généré à partir du dictionnaire de prononciation les incluant (i.e. : system5).
Globalement, les graphèmes les plus mal reconnus sont :

- **« v »** -> [F] (ɸ Nigeria) : fricative bilabiale non voisée
- **« p »** -> [p] (p Cameroun) : plosive bilabiale non voisée

A noter que le graphème et phonème « p » n'existe normalement pas en Hausa mais qu'il provient de mots étrangers.

Le graphique ci-dessous présente en abscisse les voyelles du Hausa sans distinction de longueur ou de ton et en ordonnée leur taux d'erreurs. Les voyelles sont classées selon leur fréquence d'apparition dans le corpus de test (de la plus fréquente à la moins fréquente, de la gauche vers la droite).
Nous ne présentons que les voyelles car ce sont elles qui nous ont intéressé. En effet, ce sont les voyelles qui portent les marques de tons et de longueurs dans le dictionnaire de prononciation. Ce sont donc sur celles-ci que s'est portée notre attention, dans le but d'évaluer si le codage des marques de tons et longueurs dans le lexique a un impact direct sur les performances du système de reconnaissance.

<u>Légende des différents systèmes :</u>

S3 : dictionnaire de prononciation **sans** tags

S5 : dictionnaire de prononciation **avec** tags

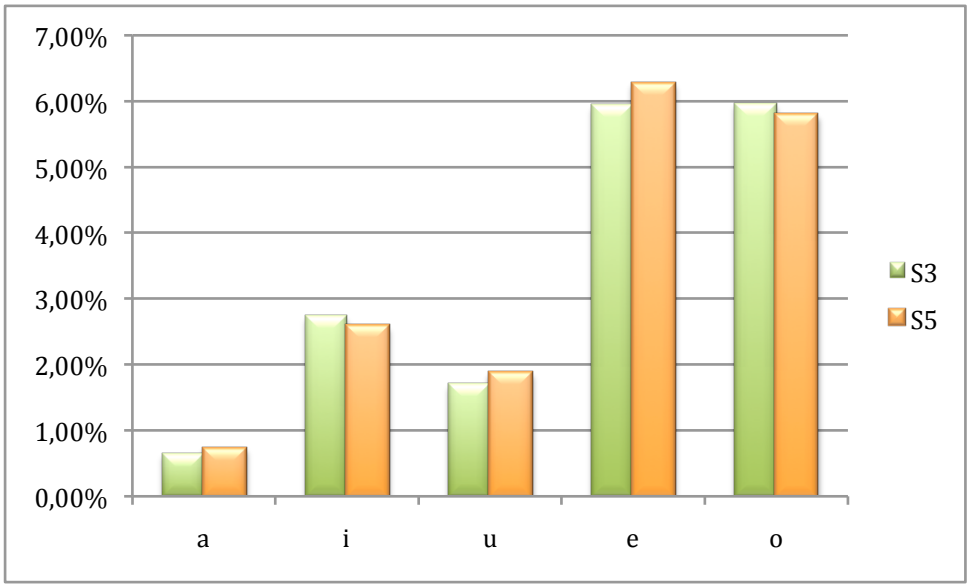


Figure 2 - Comparaison du CER des voyelles entre system3 et system5

Finalement, nous nous rendons compte à partir de l'histogramme ci-dessus que les voyelles sont plutôt bien reconnues vis-à-vis de leur nombre d'occurrence dans le corpus. Les phonèmes /i/ et /o/ sont légèrement mieux reconnus avec le modèle qui tient compte des longueurs et tons des voyelles.

Chapitre 4 – Autres travaux

Dans le cadre du projet ANR ALFFA, nous avons publié nos travaux sur GitHub[4]. GitHub est un espace de stockage de travaux collaboratifs. Cette plateforme permet de partager ses travaux à travers la toile mais aussi de pouvoir travailler à plusieurs sur un même projet, tout en conservant les versions de chaque fichier après modification. Pour cela, GitHub est basé sur le logiciel de contrôle de version Git, créé par Linus Torvalds.

Git permet de fusionner les copies de chaque projet afin de n'écraser aucune modification faite par plusieurs personnes sur un même travail. A chaque modification, une sauvegarde est faite par le biais d'un *commit* ce qui permet également de revenir en arrière si un changement non désiré a été effectué.

Git s'utilise en ligne de commande. Afin de publier nos travaux, plusieurs étapes sont nécessaires : la première consiste à initialiser un lieu de dépôt (i.e. : *repository*) via la commande `git init`. Cette commande crée un dossier .git/ qui fera le lien entre les données ajoutées, supprimées, modifiées, du dossier local et du serveur GitHub. Aussi, Git sait ce que contient le dépôt local et qui n'a pas encore été porté à son attention pour ajout au dépôt en ligne. En utilisant la commande `git status`, l'utilisateur peut vérifier ce qui a été validé ou non à un instant *t*. En effet, tant que la commande `git add` n'a pas été lancée, Git ne sait pas qu'il existe des dossiers ou fichiers à transférer sur un dépôt distant. Ainsi, pour porter un fichier ou dossier à l'attention de Git, il suffit d'utiliser la commande `git add remote origin https://github.com/nomutilisateur/nomprojet.git`. Plus précisément, cette commande permet d'ajouter un fichier ou dossier à une file d'attente. De cette manière, un fichier (ou dossier) n'est plus ignoré par Git mais aussi l'adresse indiquée lui permet de savoir qu'il existe un dépôt distant et que c'est à cet endroit qu'il doit faire les ajouts/suppressions/modifications par la suite, depuis le dépôt local. Ensuite, afin de tenir la communauté au courant des modifications apportées mais aussi pour prévenir d'une éventuelle maladresse, l'utilisateur doit consigner son travail. Pour ce faire, il faut utiliser la commande `git commit` qui permet d'enregistrer un message et de prendre un instantané du dépôt qui attribue un ID au travail avant modification. Une fois ces

[4] https://github.com/

fichiers ou dossiers portés à la connaissance de Git, ils peuvent être transférés sur le dépôt distant au moyen de la commande `git push`. Si une bévue a été commise, l'utilisateur peut toujours revenir en arrière grâce à la commande `git log`. Elle permet d'accéder à tous les *commit* et de connaître l'ID de l'état auquel l'utilisateur souhaite revenir. C'est ensuite avec la commande `git reset --hard` *`ID_du_commit`* que l'utilisateur peut restaurer le dépôt à l'instant indiqué par l'ID.

Grâce à GitHub, nous avons pu partager notre système de reconnaissance automatique du Hausa afin que d'autres puissent également travailler dessus. Nous avons aussi partagé un autre système de reconnaissance automatique, du Swahili cette fois, créé lors de travaux précédents. Ces travaux sont visibles à l'adresse suivante : https://github.com/besacier/ALFFA_PUBLIC/tree/master/ASR.

Chapitre 5 – Bilan technique et personnel

J'ai effectué ce stage car il s'inscrit dans le cadre d'un projet interdisciplinaire mêlant informatique et linguistique, mes deux domaines de prédilection.
J'ai pu appliquer mes connaissances en linguistique, acquises lors de mon précédent master en Traitement Automatique de la Langue Ecrite et Parlée. De plus, j'ai pu approfondir mes compétences en informatique acquises durant cette année de master Double Compétence en Informatique et Sciences sociales, notamment en programmation shell, ainsi qu'en Java.
Aussi, j'ai pu apprendre un nouveau langage de programmation, le python. C'est un langage doté d'une grande communauté et qui est très utilisé en traitement automatique de la langue. Cela m'a permis d'appliquer la logique apprise en cours d'algorithmique.

Lors de ces cinq mois de stage long, j'ai dû créer des scripts afin de traiter les données en Hausa qui m'étaient fournies, dans le but de les entrainer dans le moteur de reconnaissance Kaldi. J'ai utilisé le langage python car de nombreuses personnes l'utilisent aussi en traitement automatique de la langue naturelle. Il possède de nombreuses librairies utiles au traitement automatique de la langue. De plus, je savais que, si je rencontrais un bogue dans mon script, mes collègues de l'équipe pouvaient m'aider. Dans un autre langage comme Perl, par exemple, cela n'aurait pas été le cas.

Durant ce stage, j'ai appris à organiser mon temps de travail, mais aussi à partager mes idées et à dialoguer avec une communauté de chercheurs. De plus, lors de problèmes techniques, j'ai pu développer mon autonomie.

Bibliographie

Gales, M. J. (2000). Cluster adaptive training of hidden Markov models. *Speech and Audio Processing, IEEE Transactions* on, 8(4), 417-428.

Koslow, P., (1995), *Hausaland: the fortress kingdoms*. Chelsea House Publishers.

Leggetter, C. J., & Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech & Language,* 9(2), 171-185.

Povey, D., Kanevsky, D., Kingsbury, B., Ramabhadran, B., Saon, G., & Visweswariah, K. (2008, March). Boosted MMI for model and feature-space discriminative training. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference* on (pp. 4057-4060). IEEE.

Povey, D., Burget, L., Agarwal, M., Akyazi, P., Feng, K., Ghoshal, A., ... & Thomas, S. (2010, March). Subspace Gaussian mixture models for speech recognition. *In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference* on (pp. 4330-4333). IEEE.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Vesely, K. (2011, December). The Kaldi speech recognition toolkit. *In Proc. ASRU* (pp. 1-4).

Psutka, J. V. (2007, January). Benefit of maximum likelihood linear transform (MLLT) used at different levels of covariance matrices clustering in ASR systems. In *Text, Speech and Dialogue* (pp. 431-438). Springer Berlin Heidelberg.

Schlippe, T., Komgang Djomgang, E. G., Vu, N. T., Ochs, S., and Schultz, T. (2012), Hausa Large Vocabulary Continuous Speech Recognition. In *SLTU 2012*.

Schultz, T., Vu, N. T., & Schlippe, T. (2013, May). GlobalPhone: a multilingual text & speech database in 20 languages. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference* on (pp. 8126-8130). IEEE.

UNESCO (2010). Pourquoi et comment l'Afrique doit investir dans les langues africaines et l'enseignement multilingue.

Vycichl, W., (1990). Les langues tchadiques et l'origine chamitique de leur vocabulaire. In *Relations interethniques et culture matérielle dans le bassin du lac Tchad: actes du IIIème Colloque MEGA-TCHAD, Paris, ORSTOM, 11-12 septembre 1986* (Vol. 3, p. 33). IRD Editions.Webographie

Webographie

http://bargeryhausa.gotdns.com/

http://www-cs-students.stanford.edu/~blynn/gitmagic/intl/fr/ch02.html

http://www.dilaf.org/

http://doc.ubuntu-fr.org/git

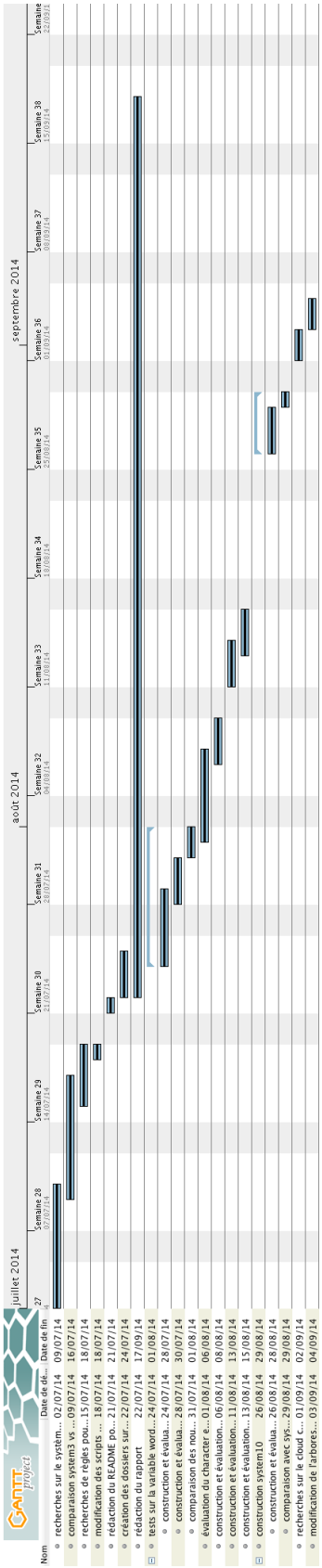
http://kaldi.sourceforge.net/

Table des illustrations

TABLEAUX

FIGURES

Annexe 1 – Diagramme de Gantt



Annexe 2 – Exemple d'un script sclite

```
#!/bin/sh

#use NIST to score (need some beautification!)

#hyp files for tri+mmi and sgmm

#hyp files for sgmm+mmi
cd /home/gauthier/kaldi/exp/system8/sgmm2_5b2_mmi_b0.1
echo "make hypotheses for SGMM+MMI"
for type in decode_it1 decode_it2 decode_it3 decode_it3.mbr decode_it4
do
    cd $type/scoring
    mkdir sclite_results/
    for i in 9 10 11 12 13 14 15 16 17 18 19 20
    do
        cat    $i.tra    |    ../../../../../utils/int2sym.pl    -f    2-
/home/gauthier/kaldi/exp/system8/sgmm2_5b2/graph/words.txt > tmp_$type-$i
        cat tmp_$type-$i | sed 's/HA[0-9]*_[0-9]* //g' > tmp_$type-line-$i
        cat tmp_$type-$i | sed 's/ [a-z]*.*//' > tmp_$type-ids-$i
        paste tmp_$type-line-$i tmp_$type-ids-$i > tmp_$type-$i_hyp
        cat  tmp_$type-$i_hyp  |  sed  's/\t/(/g'  |  sed  's/$/\)/g'  >
sclite_results/results_$type-$i.hyp  #in scoring folder
        #run sclite for calculating
        ../../../../../sclite  -r  ../../../../../data/dev/hausa_dev.ref  -h
sclite_results/results_$type-$i.hyp -i spu_id -o all
    done
    rm tmp_*
    echo "$type hyp and results done"
    cd ../../
done
```

Annexe 3 – Script python qui liste les mots erronés

```
# -*-coding:Utf-8 -*

import glob
OUT=open("wordError.txt","w")
file="decode_ref.txt"

for line in open(file):
      words = line.split(" ")
      for word in words:
            if word.isupper():
                  OUT.write(word+"\n")
            #fin if
      #fin for
#fin for
```

Annexe 4 – Script python qui liste les mots de la référence

```
# -*-coding:Utf-8 -*

import glob

file = "ref"
OUT = open("ref_lower.txt","w")
s = ""
for line in open(file):
      s=line.lower()
      OUT.write(s)
#fin for
```