Master 2 ATAL

Inférence Grammaticale

Étude de données phonétiques

Étudiants :
Carl Goubeau,
Coraline Marie et
Emmanuel Turbé

Encadrant :
Colin De La Higuera

10 janvier 2015

UNIVERSITÉ DE NANTES

Table des matières

1 Introduction

L'augmentation de l'intensité de la voix, accompagnant l'émission d'une syllabe dans un mot, est appelé "*accentuation tonique*". Ce phénomène audible est observable dans toutes les langues, mais à des degrés différents. En français par exemple, ce phénomène est quasiment imperceptible, car lors de la dictions d'une phrase, c'est le dernier mot de chaque syntagme qui porte l'accent. Dans d'autres langues, l'accentuation est plus forte et peut être placée différemment, comme en finnois, où l'accent tombe sur la première syllabe de chaque mot.

En linguistique, les chercheurs ont pris pour habitude d'attribuer un poids à chaque syllabe d'un mot. Cette pondération permet de déterminer le degré d'importance de chaque syllabe, et donc de les distinguer les unes des autres. Cependant, l'attribution des poids, n'est pas universelle, car tous les linguistes n'utilisent pas la même méthode d'attribution des poids.

Dans le cadre du cours *Inférence Grammaticale* dispensé au Master 2 ATAL de l'Université de Nantes, nous devions étudier un ensemble de données portant sur l'accentuation tonique. Cette étude se base sur un ensemble de datasets qui allient des séquences de poids à des séquences d'intonnations générées à partir de transducteurs.

Dans ce rapport, nous détaillerons l'intégralité de nos recherches sur ses données phoniques. Pour cela, nous présenterons d'abord les données que nous avons utilisées. Puis nous détaillerons les différentes approches que nous avons envisagées, avant de finir sur un ensemble d'idées d'évolutions possible du projet.

2 Présentation des données

Les données fournies sont divisées en deux parties : la première, nommée *input*, est une séquence qui contient les valeurs des poids attribués à chaque syllable. La seconde partie, que l'on appelle *output* qui est aussi une séquence, contient les différents niveaux d'accents toniques correspondant aux inputs.

Ces données ont été générées à partir de transducteurs disponibles sur le site StressTyp2. Il existe un transducteur pour chaque manière de prononcer une langue, ce qui signifie par exemple que deux transducteurs différents correspondent au français parlé en France et à celui parlé au Canada. Cependant, il est également possible que le même transducteur soit utilisé pour deux langues différentes si elles ont les mêmes propriétés d'intonation.

Pour la construction des données, des poids alétoires ont été affectés aux transitions et aux états finaux des transducteurs fournis. De plus, la complétude structurelle est assurée pour les données fournies.

Cependant ces données ne représentent pas un échantillon statistique, puisque les doublons ont été supprimés et qu'elles ont également été triées par ordre alphabétique. Pour ces raisons, les approches de machines learning basées sur les

statistiques telles que les réseaux bayésiens, ne sont pas pertinantes pour notre jeu de données.

Pour ce projet, nous n'avons pas utilisé la totalité des données fournies. Nous nous sommes également rendu compte que certains fichiers comportaient des erreurs, comme par exemple une inversion entre les inputs et les outputs.

3 Nos approches

3.1 Weka

Notre première approche, fût de tenter d'utiliser Weka, dans le but de clusteriser l'ensemble des données fournies. Les informations manipulées correspondent aux valeurs présentes dans une fenêtre située autour de la syllabe courante : les deux syllabes précédentes et les deux syllabes suivantes. Malheureusement, le clustering n'excède pas les 70% de précision, quelque soit l'algorithme testé. Il est cependant fort probable que l'ajout de d'autres attributs permette d'augmenter les résultats.

La combinaison des ensembles de données baisse grandement les résultats. Ceci est principalement dû au fait que chaque transducteur corresponde à une manière de parler diffèrente. Ce qui peut varier énormément d'une langue à l'autre. Il en va donc de même pour la position des intonations, qui varie également plus ou moins fortement en fonction de la langue.

Un second essai, fût d'utiliser la classification par *table de décision*, et d'étudier les règles ainsi générées. Pour commencer, de premiers tests ont été effectués fichier par fichier, ce qui a permis d'obtenir des résultats proche des 50% de précision. Dans un second test, nous avons voulu savoir, s'il était possible de déterminer la position des accents tonique les plus forts, plutôt que de rechercher la structure complète des intonnations. Nous conservons les accents *"s2"* pour signaler l'accent principal, et notons le reste (s0, s1) avec *"s0"* pour désigner une syllabe qui ne contient pas l'accentuation principale (Figure 1).

Une telle approche [1] nous à permis d'augmenter significativement les résultats, et cela, même en faisant de la classification [2] sur l'ensemble complet de données. Avec les 270 règles trouvées par Weka, la précision augmente à 86.5%.

fichier	précision
English (#154)	0.750
Final (#111)	0.907
Initial (#112)	0.786
Tous les fichier	0.865

Table 1 – Précision de la classification de syllabes : accent principal ou non

1. script de transformation des données disponible sur GitHub https://github.com/Slayerxoxo/GI_Project/blob/master/primary-stress-to-arff.py
2. Classify, DecisionTable, RandomSearch -P 1,2,3,4,5,6 -seed 2 cross-val 10

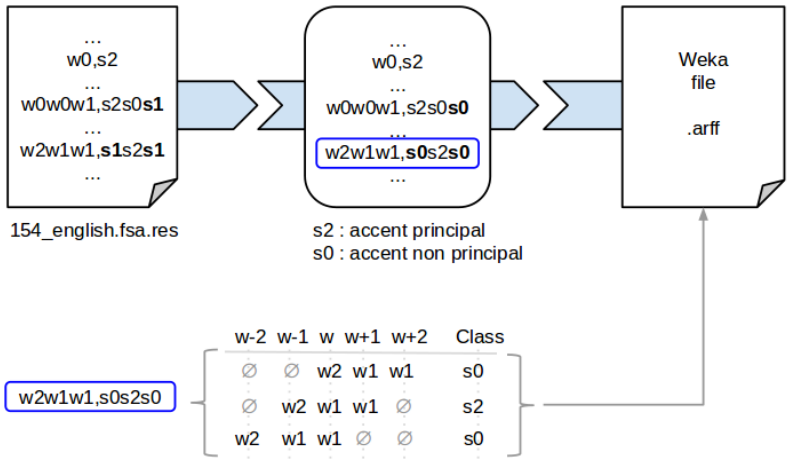

FIGURE 1 – Transformation des données pour Weka

3.2 Apprentissage par l'algorithme A_k-TSS

3.2.1 Rappel sur l'algorithme A_k-TSS

La seconde approche que nous avons eu concerne l'utilisation d'un algorithme d'apprentissage : le A_k-TSS. Cet algorithme, que nous avons étudié en cours [de la Higuera, 2014], permet de construire un automate à partir d'un langage k-testable. Cet algorithme à été introduit par P. García & E. Vidal dans l'article *Inference of K-testable languages in the strict sense and applications to syntactic pattern recognition* [García et Vidal, 1990]

Un langage k-testable L, est un langage dont l'appartenance d'un mot w à L, dépend du préfixe, du suffixe et de l'ensemble des facteurs de w de longueur k. Un langage est donc localement testable si il est k-testable pour une fenêtre donnée de taille k.

3.2.2 Utilisation de l'algorithme A_k-TSS

Comme il l'a déjà été expliqué lors de la présentation des données, les datasets utilisés associent des séquences de poids à des séquences d'intonations par l'intermédiaires de transducteurs. Dans cette approche nous avons utilisé l'algorithme A_k-TSS dans le but de reconstruire plusieurs automates.

Pour ce faire, nous avons réutilisé un script en Python, créé par Hugo Mougard & Gregoire Jadi en 2011 [Mougard et Jadi, 2011] à partir du livre Gramatical Inference [de la Higuera, 2010], qui décrit l'algorithme A_k-TSS. Nous avons également légèrement modifié ce programme, afin qu'il puisse prendre les datasets comme données d'apprentissage. Dans l'intégralité de nos tests, la taille du k est de 3.

Les trois tests suivants seront illustrés par l'exemple de la langue Kawaiisu [3].

Traitement des inputs

Les inputs sont les séquences de poids contenus dans les datasets. Pour le premier test, nous avons passé les inputs comme données d'apprentissages du script, ce qui nous donne un automate pour chaque dataset. Les automates produits par le script, et donc par l'algortithme A_k-TSS sont des automates déterministes assez complexes. Ils contiennent de nombreux états (cf FIGURE 2).

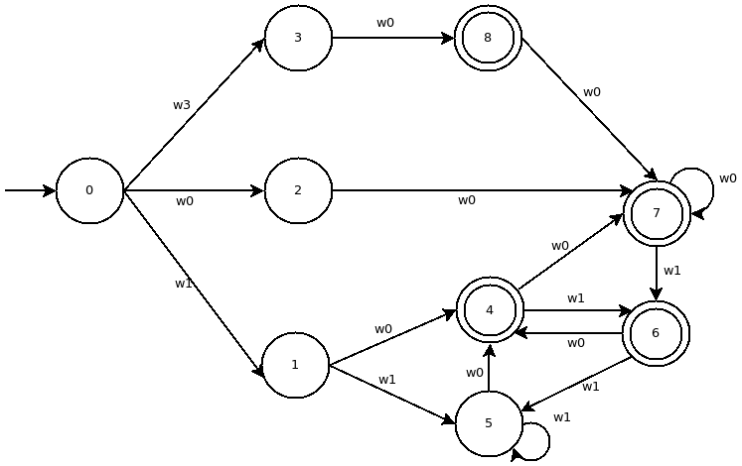

FIGURE 2 – Automate généré par A_3-TSS pour les séquences inputs de la Langue Kawaiisu

L'automate généré nous permet de dire, pour chaque langue, si une séquence de poids de syllable est acceptée, c'est-à-dire si elle existe dans la langue.

Au terme de cette expérience, peu de conclusions peuvent être tirées. En effet, les automates obtenus en sortie ne sont pas comparables, car les langues qu'ils représentent sont différentes tous comme la méthode d'attribution des poids.

Traitement des outputs

Les outputs sont les séquences d'intonations contenus dans les datasets. Pour le second test, nous avons remplacé les données d'apprentissages précédentes par les outputs. Encore une fois, les automates produits en sortie sont déterministes et complexes. Comme pour les inputs, ils contiennent de nombreux états (cf FIGURE 3).

L'automate ainsi généré permet, pour une séquence d'intonations, de savoir si la séquence est acceptée. En d'autres termes, il est envisageable de faire de

3. Le Kawaiisu est une langue uto-aztèque, du Nord de la branche des langues numiques parlée aux États-Unis, dans le sud de la Californie.

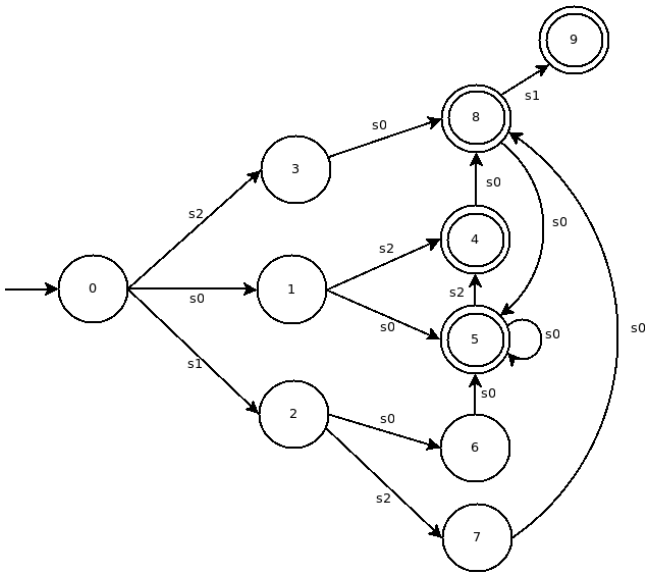

FIGURE 3 – Automate généré par A_3-TSS pour les séquences outputs de la Langue Kawaiisu

la détection de langue à partir de ces automates si les données d'accentuation tonique peuvent être fournies.

Au terme de cette seconde expérience, les même conclusions peuvent être faites que pour les inputs, à savoir que les automates obtenus en sortie ne sont pas comparables entre eux.

Traitement des inputs/outputs

Le dernier test que nous avons effectué avec l'algortithme A_k-TSS concerne l'ensemble des données des datasets. En effet, pour ce test nous avons placé l'association input/output comme ensemble de données d'apprentissage du programme, ce qui nous a permis d'obtenir de nouveaux automates (cf FIGURE 4). Les transitions deviennent donc un couple poids/intonation.

Les automates obtenus sont équivalents aux transducteurs utilisés pour générer les datasets. Ils sont équivalent car pour une séquence d'entrée fournie si on parcours l'automate sans tenir compte de l'accentuation dans la transition, un seul chemin est possible et ce chemin correspond à l'accentuation. Cependant ces automates sont assez complexes car ils contiennent beaucoup d'états et plusieurs états finaux. Ils sont également déterministes car pour chaque état, une seule transition est possibles pour la même combinaison poids/tonique. Les automates produits n'apportent pas plus d'informations sur les données.

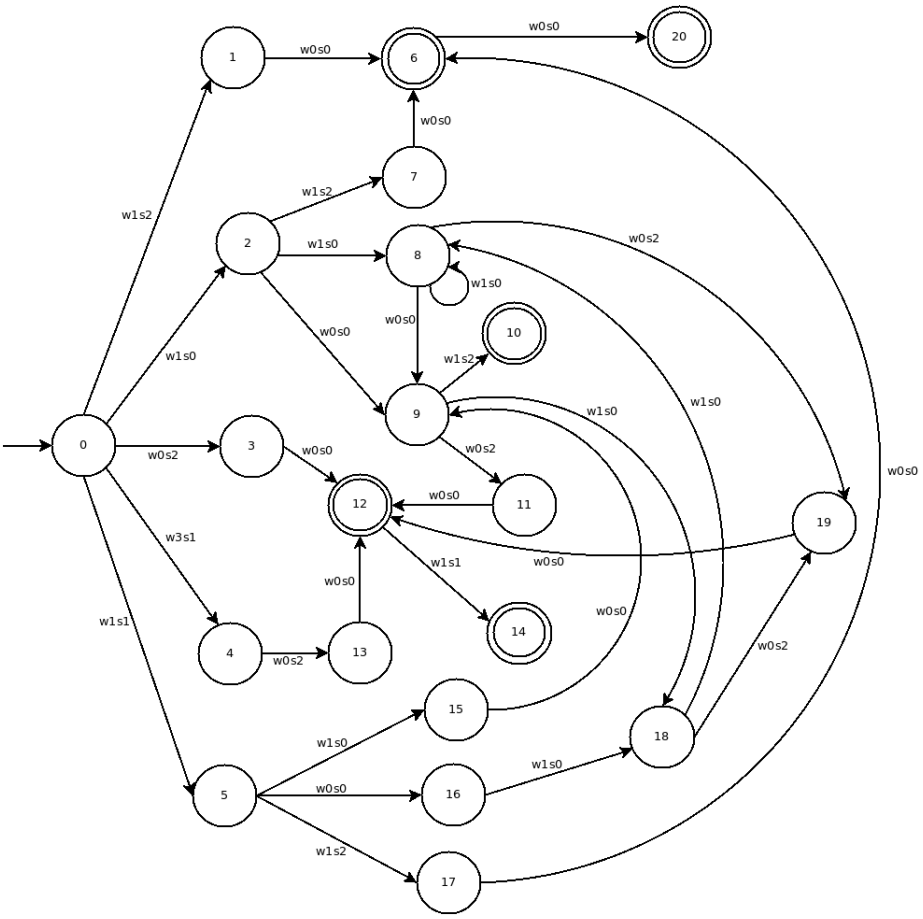

FIGURE 4 – Automate généré par A_3-TSS pour l'ensemble des séquences de la Langue Kawaiisu

3.2.3 Conclusion

En définitive, ces différentes expériences ne nous ont malheureusement pas appris beaucoup de choses sur les données, si ce n'est que nous sommes capable, par l'intermédiaire de l'algorithme A_k-TSS, de construire des automates équivalents aux transducteurs utilisé sur les datasets. Ainsi, nous pouvons désormais nous demander si d'autres algorithmes d'apprentissages sont également capable de générer ces transducteurs, et de nous apporter plus d'informations sur les datasets.

4 Pour aller plus loin

Dans cette partie se trouve toutes les idées non implémentées pour les données. Il n'y a donc ni code, ni tests disponible pour cette section, mais seulement des réflexions.

4.1 L'algorithme $L*$

L'algorithme $L*$ [Angluin, 1987] apprend un DFA minimal pour un langage régulier donné. Cet algorithme d'apprentissage ne connait initialement rien à propos du langage L. Il tente d'apprendre un DFA A tel que $L(A) = L$. Pour cela, il envoi des requêtes d'appartenance et d'équivalence à un *Oracle* :

— Une requête d'appartenance consiste à demander si une chaîne appartient au langage L.

— Une requête d'équivalence consiste à demander si le DFA construit est correcte. Si *l'Oracle* répond négativement, il fournit une chaîne contre-exemple.

Dans le soucis de tester cette hypothèse, nous avons cherchés à implémenter cette méthode. Cependant, nous n'avions pas de moyens rapide et automatique pour mettre en place l'Oracle. Par conséquent, nous n'avons pas pu coder cette partie, ni pu aller au terme de cette idée. Comme avec l'algorithme A_k-TSS, nous aurions aimé pouvoir générer des DFA à partir des ensembles de données fournies, en se servant de l'algorithme $L*$.

4.2 Game Theory

Comme présenté dans le sujet, il doit être possible d'adapter les transducteurs en automates en servant des idées issue de *game theory*. Ici les transitions seraient le poids des syllables et les états seraient associés aux niveau d'accent tonique résultant.

L'automate apprit ne sera pas déterministe et l'apprentissage serait éffectué à partir des chaînes déjà fournies. On se retrouve dans une configuration différente de $L*$ dans le sens où l'on a déjà les données acceptées sans avoir à consulter un oracle. On construit un modèle consistant avec les données. Deuxième différence, nous n'avons pas la possibilité de savoir si une chaîne n'est pas acceptée. Il est donc possible que lors de la construction de l'automate, qu'un cas non présent dans les données d'apprentissage soit accepté par généralisation.

4.3 Comparaison avec HTM

Cette partie ne concerne pas directement les données fournies mais plutôt les données qui ont permi de générer les transducteurs. Les données sont des séquences dont les propriétés peuvent être apprises. L'algorithme d'apprentissage *online, Cortical learning algorithm*, basé sur Hierarchical temporal memory est censé être performant sur ce type de données. Les principes de l'algorithmes sont décrits dans le livre de Jeff Hawkins [Hawkins et Blakeslee, 2007]. L'algortithme étant basé sur le fonctionnement du néocortex, qui est une partie du cerveau, il serait possible de comparer l'évolution de l'apprentissage avec ce que l'on sait de l'apprentissage chez l'humain.

5 Conclusion

Cette étude de données phoniques fut un projet fort instructif, mais également assez difficile à réaliser. Durant ce travail, de nombreuses difficultées nous

ont ralentit. La première fut de déterminer précisement quel était le travail à faire, et quel serait le contenu du projet. Le sujet proposé étant assez vague, il nous a fallu du temps pour déterminer précisément ce que nous pouvions faire avec les données.

Comme tout travail de recherche, plusieurs pistes étaient envisageables. Cependant, il était difficile de déterminer lesquelles seraient les plus pertinentes. Nos premières idées étaient de tenter une approche statistique, mais aux vues des données disponibles, ce n'était pas envisageable. C'est pourquoi nous avons songer à deux autres approches : une première Weka, et une seconde avec l'utilisation d'algorithmes d'apprentissages.

Au fur et à mesure de nos études, d'autre idées nous sont venue, comme par exemple l'utilisation d'autres algorithmes. Malheureusement toutes ces pistes n'étaient pas testable, compte tenu du peu de temps qu'il nous restait.

Malgré ces difficultés, ce projet nous a quand même permis d'apprendre. Ce fut notamment une nouvelle expérience de recherche. Ce travail nous a également permis de mettre en pratique d'autres connaissances aquises durant notre master (programmation, recherche, lecture d'articles, calculs, ...), et de nous familiariser avec les concepts d'Inférence Grammaticale.

Les script utilisés dans ce projet sont disponibles sur le dépôt GitHub `https://github.com/Slayerxoxo/GI_Project`.

Références

[Angluin, 1987] Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106.

[de la Higuera, 2010] de la Higuera, C. (2010). *Grammatical Inference : Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA.

[de la Higuera, 2014] de la Higuera, C. (2014). Grammatical inference : learning from text. http://madoc.univ-nantes.fr/mod/resource/view.php?id=212773. [Ressource en ligne disponible au 09 Janvier 2015].

[García et Vidal, 1990] García, P. et Vidal, E. (1990). Inference of k-testable languages in the strict sense and application to syntactic pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(9):920–925.

[Hawkins et Blakeslee, 2007] Hawkins, J. et Blakeslee, S. (2007). *On intelligence*. Macmillan.

[Mougard et Jadi, 2011] Mougard, H. et Jadi, G. (2011). Implementation of various simple algorithms of grammatical inference. https://github.com/m09/Grammatical-inference. [Ressource en ligne disponible au 09 Janvier 2015].