Word-spotting pour indexage des fichiers audio

Johan Jansen

Indexage des fichiers audio

Index

1. Introduction

Le but de ce travail est de trouver un moyen d'indexer des enregistrements des cours à l'Université de Genève. Il s'agit ici d'un nombre élevé d'enregistrements (10'000 au moins) avec des qualités sonores très différentes. Les cours furnt enregistrés d'abord avec des microphones fixes sur des cassettes avec un qualité médiocre, surtout quand le locuteur se promènait beaucoup pendant le cours. Ensuite des microphones fixes de meilleures qualités furent utilisés et le support digital améliarat encore la qualité. Finalement les microphones "cravatte" donnent une liberté totale au locuteur avec toujours une qualité quasi parfaite. Cette dernière amélioration garantit également l'indépendance de l'acoustique de la salle dans laquelle le cours a lieu.

La première étape consiste à établir une liste des fréquences des mots utilisés dans le cours, comparer cette liste avec les fréquences des mots dans le corpus de la langue et en extraire des mots clés spécifiques du cours. En même temps les mots clés "standards", spécifiées d'avance, seront vérifiés.
Pour cette première étape des méthodes différents peuvent être appliquées, qui ont tous prouvé leur efficacité dans leur domaine, transcription complète de texte parlé dans l'environnement médical, word-spotting (recherche des mots clès en direct) dans les Call-Centers des grandes entreprise et pendant des grandes conférences avec transcription et indexage quasi en temps réel.

2. Méthodes disponibles

Actuellemen trois méthodes sont disponibles pour la recherche des mots clès dans des fichiers audio/vdéo, chaque méthode avec des variantes:

1. Transcription automatique complète de tout le texte parlé suivi d'un indexage standard du fichier transcrit avec recheche des mots clé.

Avantage: Disponible avec un taux de réussite autour de 85% et une réjection fausse des

mots cherchés de 5%.
Disavantage: Traitement long, pas beaucoup pus rapide que le texte parlé.

2. Recherche des mots qui correspondent plus ou moins au mots clés recherchés pendant la première analyse du texte, soit juste après le premier assemblage des phonèmes, soit après la vérification du mot dans les dictionnaires. Dans le premier cas il faut faire la vérification soi-même, puis comparer le résultat avec le mot clé recherché. Ceci peut faire gagner du temps de traitement.

Avantage: Rapidité, des facteurs plus grand que cent peuvent être obtenus par rapport à la durée du texte parlé.
Desavantage: Taux de réussite médiocre dans la détermination du mot sans utiliser le contexte de la phrase.

3. Recherche de correspondances des phonèmes entre les mots dans le texte parlé et les mots clés parlés.

Avantages: Taux de réussite élevé et rapidité extrême, des facteurs au-délà de 1000 sont possibles.
Desavantages: Difficile à appliquer avec des fichiers audio multi-locuteur ou avec les fichiers audio archivés de longue date.

Dans le cas oú le langage parlé n'est pas encore connu seulement les méthodes 1 et 3 sont applicables, éventuellement un bout de 1 pour la détection de la langue suivi de 2 ou 3 pour la recherche des mots.
Une solutions combinée de 1 et 2 avec une analyse complète de la phrase seulement si dans cette phrase un mot est trouvé qui pourrait correspondre à un mot clé.

2.1 Transcription automatique complète.

Des programmes de transcription automatique complète sont actuellement en service dans tous les domaines avec plus ou moins de réussite avec des taux d'erreur dans les mots de 5 à 15%.
Pour indexage un taux d'erreur de transcription plus petit que 30% est probablement suffisant.

La mise en route d'un système d'indexage basé sur un de ces programmes existants ne nécessite 'que' la mise en route d'un détecteur de mots clés dans le fichier de texte transcrit.

Etant donné que la transcription est déjà vérifiée contre une dictionnaire complète de la langue utilisée en principe aucune autre vérification est nécessaire.
Il suffit de comparer une liste de tous les variants possibles des mots clés recherchés avec les mots transcrits. Pour obtenir confiance dans les résultats il faut quand-même faire une analyse des mots non-transcrits ou mal-transcrits.
En principe le travail peut commencer immédiatement.

2.2 Recherche des mots clés sans transcription complète.

Dans ce cas la chaîne de transcription est interrompue aprés l'assemblage du mot à partir des phonèmes, donc sans prendre en considération le contexte du mot dans la phrase. Par contre, le mot est vérifié contre des dictionnaires, mais sans connaissance du contexte le système ne peut pas distinguer homonymes et homophones et le taux d'erreur rest élévé. Cette méthode de word-spotting

est extensivement utilisé dans les call-centers en temps réel et en temps différé.
Pour gagner en efficacité les programmes de word-spotting ne terminent même plus la construction du mot si ceci ne peut plus correspondre à aucun des mots clés recherchés.
Par contre dans ce cas il n'est plus possible d'analyser les mots dans leur contexte, même si un mot clé est trouvé et ainsi réduire le taux d'erreur sans refaire tous le mots dans la phrase.
Il existent des programmes de transcripion automatique contiennent un API qui permet de prendre les mots juste après leur construction isolée.

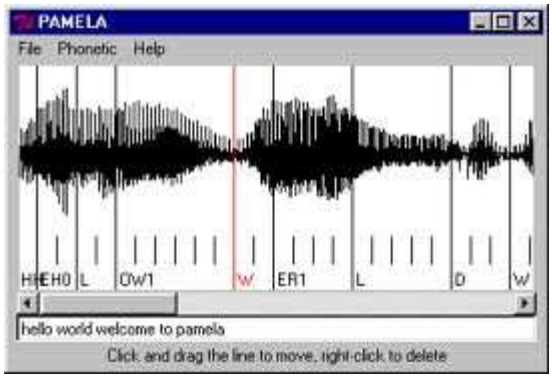
2.3 Recherche des mots clés au niveau des phonèmes.

Bien que cette recherche soit extrêmement rapide, l'application est impossible dans un environnement de fichiers audio de plusieurs locuteurs, dont la plus grand nombre n'est plus présent.
Ce système exige que le locuteur lui-même dicte les mots clés au moins deux fois pour obtenir un taux de réussite élevé à une vitesse très élevée.

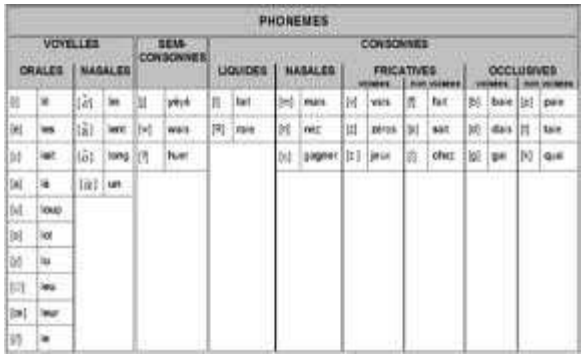
3. Reconnaissance de voix, Automatic Speech Recognition(ASR)

Un système de reconnaissance vocale consiste essentiellement de trois parties:

1. La partie accoustique, le traitement du son, l'élimination des bruits, puis le découpage des ensembles de sons qui constituent le mot en phonèmes. En Français on distingue 16 voyelles, 3 semi-consonnes, 20 consonnes et 3 cas particuliers. Il existent des modèles accoustiques pour différents environnements; téléphone, micro pc, micro profesionnelle de salle de cours ou de congres ou micro cravatte. Ce grand nombre de cas différents est la plus difficile à suivre dans la reconnaissance vocale, mais les programmes modernes sont pourvus de possibilités d'apprentissage extrêmement performant.



Hello world



2. La partie qui reconstitue le mot à partir des phonèmes. Le modèle de Markov caché, HMM, est utilisé ensemble avec des dictionnaires et vérificateurs d'orthographe, qui examinent tous les conjugaisons possibles du mot our trouver le mot le plus vraisemblable.

3. Cette partie, qui, dans les programmes modernes, travaille en combinaison avec la deuxième partie reconstruit des parties du discours, également avec des HMM et utilisent en plus des méthodes de POS tagging utilisant l'algorithme de Viterbi pour obtenir la partie de discours la plus probable. Une la solution la plus probable trouvée, les programmes modernes font un retour à la partie 2 pour optimiser uncore la solution. Ceci évidement avec beaucoup d'effort de calcul.

Word-spotting pour indexage des fichiers audio Johan Jansen

4. Programmes disponibles.

En dehors des essais de lecture et transcription des fichiers faits par des sociétés externes, comme Virage de Autonomy et Media Mining Indexer de Sails Labs Technology, deux programmes ont été pris en considération pour des essais:

1. Sphinx-4 en distribution libre de SourceForge .Le programme consiste de parties bien séparées, mais est difficile à installer et à paramétrer. Des problèmes sérieux d'incompatibilité de la dernière version et mes version de java JDK6 et eclipse 1.3 et Hélios m'ont empêché de l'installer correctement. A revoir.
2. Dragon NaturallySpeaking, le programme initialement développé par IBM depuis 1975, mais qui vit maintenant sa propre existance. Initialement distribué par la société Scansoft, il est actuellement vendu par Nuance en 6 langues. J'ai installé la version française, gracieusement mis à disposition par le service informatique de l'Université.

Le programme utilise une sorte de recherche de mots-clés pour pouvoir effectuer des commandes dites par l'utilisateur qui sont exécutés par le programme, en principe l'utilisateur n'aura plus besoin de clavier.
L'apprentissage est assez longue, ce qui peut s'expliquer par l'utilisation prévu pour une seule personne. Une fois l'apprentissage initial terminé le programme ne cesse de s'améliorer avec les données reçues au cours de son utilisation.

5. Essais

Comme essai de transcription, deux cours, un de l'année 72-73 et un de 82, ont été utilisés.
D'abord j'ai entraîné le programme avec ma voix, en lisant à haute voix un texte de 5 minutes proposé par le proramme. La transcription ensuite d'autre textes lus par moi à haute voix était assez surprenant, quasiment parfait.
Par contre, la transcription du cours était catastrophale, rien que des 'le', 'que' ou 'de'.
Ensuite j'ai utilisé le cours même pour apprentissage pendant une heure, le résultat de la transcription du même texte à la suite était d'excellente qualité et comprenait de seulement quelques pourcent d'erreurs. Le cours était parlé avec une voix très claire et distinctive.
L'autre cours était d'une qualité nettement moindre, parlé avec une voix très platte avec beaucoup de 'uh' et de 'mm' et avec des bruits de papiers comme fond. Le même procédé d'apprentissage et la transcription séquentielle avec le même cours donnait presque le même résultat que le premier cours, cette fois 7 à 8 pourcent d'erreurs.
Un problème n'est pas encore résolu, le programme s'arrête après 30k de taille de texte, probablement parce que je l'utilise en version essai.

Pour obtenir un premier estimation de l'utilité de la transcription, j'ai écrit un programme qui lit le fichier transcrit, construit fait des tables des mots avec leur fréquences(vérifie la loi de Zipf), ensuite enlève les mots inutiles(stopwords) et sort une liste des mots tries alphabétiquement pour trouver les mots utiles pour indexage.

6. Proposition de travail.

Par le fait que le temps de traitement nécesaire pour l'indexage n'est pas un critère prédominant, la manière de procéder me semble la suivante:

Word-spotting pour indexage des fichiers audio Johan Jansen

1. Décider quel programme de transcription automatique à utiliser, Sphinx, Dragon ou Autre.
   Les critères sont:
   Version pour la langue française disponible.
   SDK ou API disponible pour dictionnaires privés supplémentaires dans la transcription.
   Eventuellement avec accès au mots avant la reconstruction complète de la phrase.

2. Passer quelques fichiers avec transcription complète.
   Un jeu de fichiers avec des qualités différentes est fourni par le SI.
   Pour transcription Dragon NaturallySpeaking et Sphinx seront utilisés.
   Avec comptage de tous les mots trouvés
   Etablir le temps de traitement moyen et le taux d'erreur.
   Vérifier la table des mots et leurs fréquences ainsi obtenue pour obtenir les distributions des variations possibles des mots.
   Comparer la distribution des fréquences ainsi obtenues avec les fréquences dans un corpus "standard" de la langue.
   Extraire les mosts plus fréquents dans les fichiers audio.
   Rapport concernant les résultats obtenus.

3. Décider si ce modèle peut fonctionner avec un taux d'erreur acceptable
4. Etablir la liste des mots clés, ajouter des mots au dictionnaire de transcription si nécessaire.
5. Développer et tester le programme de détection des mots clés dans le texte transcrit.
6. Repasser les fichiers avec le programme de recherche des mots clés.
7. Rapport des résultats, encore à faire dès que la liste actuelle des mots est analysée.

Etat actuel:

J'ai pu tester (un peu) Dragon NaturallySpeaking en version française, les résultats des transcriptions est tout à fait acceptable, avec largement en dessous de 10% d'erreurs.
Je n'ai pas eu accès au SDK de Dragon et n'a pas pu le tester.
Le programme doit être vérifié au niveau de word spotting, il semble possible de créer ses propres commandes, donc une liste des mot-clés doit être possible de créer.
Je n'ai pas pu vérifier le résultat de Sphinx-4, version française.

Les deux fichiers transcrits ont ét analyse sur leur contenu, d'abord un simple comptage des mots et leur fréquences en fonction des fréquences, puis une sélection des mots qui contient la plupart des mots intéressants en ordre alphabétique. Dans les deux textes les pourcentages des mots les plus fréquents et non-distinctifs est de 1 à 3%, mais contiennent 30% des mots dans le texte.
A l'autre coté, le nombre de mots qui n'apparaissent qu'une fois dans le texte est 66%, mais contiennent que 20% des mots dans le texte.

Annexe: Exemple algorithme de Viterbi (Wikipedia)

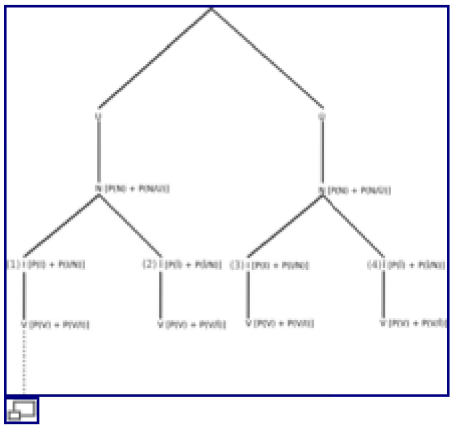
Imaginons un canal bruité qui supprime tous les accents d'un texte. À partir du message *m* : « Université » on observe le message *o* : « Universite ».

Plusieurs messages ont pu conduire à cette observation :

- Ùnîvêrsîtê
- Ùnîvêrsîtè
- Ùnîvêrsîté
- …
- Université
- …

On peut connaître la probabilité de chaque lettre en s'intéressant à sa probabilité d'apparition dans la langue française et l'on peut affiner cette probabilité en cherchant la probabilité qu'une lettre apparaisse alors qu'elle est précédée d'une autre (et ainsi de suite), on parle d'unigrammes, de bigrammes, trigrammes…

On construit l'arbre suivant (avec les bigrammes, P(X) correspond à la probabilité de voir apparaître X, P(Y/X) est la probabilité d'avoir Y sachant que l'on a eu X -- cf image).



Déroulement de l'algorithme de Viterbi sur un exemple

La probabilité d'une branche est le produit de la probabilité de tous ses nœuds. À chaque nœud, on considère la probabilité de l'unigramme (P(X)) et du bigramme (P(X/Y)). Dans la pratique ces probabilités sont pondérées et normalisées et l'on obtient pour chaque nœud la valeur $\lambda_1 P(X) + \lambda_2 P(Y / X) + \lambda_3 P(Z / XY) + ...$ avec $\lambda_0 + \lambda_1 + \lambda_2 + ... + \lambda_n = 1$ (*n* correspondant au degré de n-gramme désiré), puisque l'on s'intéresse à une probabilité, donc comprise entre zéro et un (pour l'exemple, les valeurs $\lambda_1 = 0.9$ et $\lambda_2 = 0.1$ semblent optimales).

En regardant l'arbre, on se rend compte rapidement que les probabilités du troisième étage (pour le caractère observé « i ») sont identiques et ne dépendent que de l'étage précédent. De même, les sous-arbres de ces nœuds seront identiques, *il est donc inutile de dérouler les sous-arbres sous les branches de plus faible probabilité puisque l'on est sûr que la probabilité totale des branches engendrées sera plus faible*.

Par exemple, supposons que nous ayons obtenu comme probabilités au troisième étage de notre arbre :

- P(UNI) = 0.8
- P(UNÎ) = 0.15

- P(ÙNI) = 0.04
- P(ÙNÎ) = 0.01

Les sous-arbres en dessous du I de UNI et en dessous du I de ÙNI sont strictement équivalents, de même pour les sous-arbres en dessous du Î de UNÎ et du Î de ÙNÎ. On aura donc les probabilités suivantes :

- P(UNI…) = 0.8 * P(Sous-Arbre(NI)) *(1)*
- P(ÙNI…) = 0.04 * P(Sous-Arbre(NI)) *(3)*

et

- P(UNÎ…) = 0.15 * P(Sous-Arbre(NÎ)) *(2)*
- P(ÙNÎ…) = 0.01 * P(Sous-Arbre(NÎ)) *(4)*

Quelle que soit la probabilité de chacun des sous-arbres, on se rend compte que certains seront plus faibles que d'autres rapidement, par exemple la branche *(4)* sera inférieure à la branche *(2)* et la branche *(3)* inférieure à la branche *(1)*. On peut donc élaguer l'arbre en les supprimant et continuer le calcul sur les branches restantes (suppression de la moitié des possibilités à chaque étape).

Références

Beaucoup d'articles sur la reconnaissance vocale et transcription automatique ont été écrits à la fin des années 90, depuis le nombre d'articles a bien diminué et se limitent souvent à des domaines très spécifiques. Il semble que la plupart des solutions possibles ont été explorés sans vraiment obtenir des taux de réussite plus élevés.

Les principe de base du traitement de langue parlée sont très bien expliqués dans:

Speech and Language Processing, Second Edition, Daniel Jurafsky et James H. Martin, Pearson international edition, 2009, New Jersey,USA

Un article général qui pose bien les problèmes, contient un bibliographie interessant, et dont j'ai tiré quelques chiffres est:

Auto Indexing: What has been accompished and the road ahead, Ivan Magrin-Chagnolleau et Nathalie Parlangeau-Vallès, 2001,CNRS,France.

Un article des années 90 qui aborde les méthodes de word-spotting dans les fichiers audio:

Fast implementation methods for Viterbi-based word-spotting, K. M. Knill et S. J. Young, Proceedings ICASSP-96, Atlanta, GA, Etats-Unis

Un article plus récent abordant différent méthodes de détection des mots clés dans des large bases de données audio est:

An application of recurrent neural networks to discrimative keyword spottig, Santiago fernandez, Alex Graves et Jürgen Schmidthuber, 2007,Lugano, Suisse.

Références au sujet du modéle de Markov caché et l'algorithme de Viterbi (Wikipedia):

- *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, A Viterbi - IEEE Transactions on Information Theory, 1967. (papier fondateur de Viterbi)
- *The Viterbi algorithm*, GD Forney Jr - Proceedings of the IEEE, 1973.
- *The Viterbi algorithm as an aid in text recognition* (Corresp.) D Neuhoff - IEEE Transactions on Information Theory, 1975
- Lawrence R. Rabiner (February 1989). "A tutorial on Hidden Markov Models and selected applications in speech recognition". *Proceedings of the IEEE* **77** (2): 257–286, février 1989.
- James K. Baker (1975). "The DRAGON System -- An Overview". *IEEE Transactions on Acoustics Speech and Signal Processing* **23** (1): 24–29. doi:10.1109/TASSP.1975.1162650.
- Frederick Jelinek, Lalit Bahl, Robert Mercer (1975). *IEEE Transactions on Information Theory* **21**.
- Xuedong Huang, M. Jack, and Y. Ariki (1990). *Hidden Markov Models for Speech Recognition*. Edinburgh University Press. ISBN 0748601627.