

LLia_Phon : Guide pour développeur

Roger Mampey

April 3, 2004

1 Introduction

Ce document s'efforce de définir, après coup, les spécifications de la synthèse vocale `LliaPhon`. Il s'agit d'une réécriture sous la forme d'un seul exécutable de la synthèse `LiaPhon` proposée par Frédéric Béchet sous licence GPL, sous la forme d'un ensemble de processus Unix communiquant par pipe. Frédéric Béchet fournit une description des fonctionnalités de la synthèse [1], mais s'adresse implicitement à des gens qui connaissent déjà quelque peu le domaine.

Dans `LliaPhon` (le premier L est pour "light"), comme dans `LiaPhon`, la synthèse enchaîne 4 opérations successives : formattage, étiquetage, phonétisation et prosodie. Pour avoir une idée préalable de ce que sont ces opérations, il est préférable de consulter l'annexe **Traitements linguistiques**. Dans ce document correspond, dans ce qui suit, un chapitre à chacune de ces 4 opérations.

Dans `LiaPhon`, chaque opération est réalisée par un ou plusieurs processus. Dans `LliaPhon`, le coeur du programme est une boucle qui lit une chaîne de caractères, la transforme en une liste d'items successivement traitée par chacun des modules mettant en oeuvre une des 4 opérations.

Plus précisément, l'objet qui est passé en argument à tous les modules est une liste doublement chaînée (une structure finement dénommée `liste`) d'items (une structure fourre-tout dénommée `item` contenant les éléments construits au fur et à mesure par les 4 opérations).

Pour ceux à qui viendrait l'idée de décortiquer le code de plus près, les structures `liste` et `item` sont définies dans `util.h` et les moyens de les manipuler dans `util.c`. La boucle principale et les appels successifs aux différents modules sont dans `synthese.c`. Les modules dans les fichiers `nett.c`, `tagger.c` et `phonet.c` correspondant aux 3 premières grandes phases de formattage, étiquetage et phonétisation (le calcul de la prosodie - très élémentaire - est resté dans `synthese.c`).

Au départ, on procède à une segmentation de la chaîne d'entrée. issue d'un fichier ou de l'entrée standard, en utilisant comme séparateurs les caractères ' ', \t, \r et \n (blanc, tabulation, retour chariot et nouvelle ligne). Chaque item contient alors une

chaîne de caractères (attribut 'ch' qui est une chaîne de caractères jamais nulle et jamais vide) et le ou les séparateurs qui suivaient cette chaîne dans le texte initial (attribut 'sep' normalement ici jamais nul, mais qui peut l'être pour des items apparaissant plus tard)

On fait les hypothèses suivantes sur la chaîne d'entrée :

- C'est une chaîne de caractères codés sur 8 bits.
- Il n'y a pas de caractères "de contrôle". Les seuls caractères sans graphie correspondante sont les séparateurs standard (`\r`, `\n`, `\t`).
- Il n'y a pas de balises - html ou autres - ou de mot-clés. La prise en compte de ce type d'information ressort normalement des pré-traitements. Ceci dit, la distinction n'est pas forcément si claire : dans les mails, le traitement de l'entête ressort bien du pré-traitement mais on trouve très souvent des adresses mails ou des url dans le corps du texte.

Et maintenant un exemple de segmentation initiale.

A poursuivre une idée fixe, on ne risque pas d'aller bien loin.

devient

```
( [A| ] [poursuivre| ] [une| ] [idée| ] [fixe,| ]  
  [on| ] [ne| ] [risque| ] [pas| ] [d'aller| ] [bien| ] [loin.|\\n]  
)
```

Par la suite, si la valeur des séparateurs n'est pas significative, une telle liste sera plutôt écrite :

```
(A poursuivre une idée fixe, on ne risque pas d'aller bien loin.)
```

2 Formattage

OBJECTIF : Transformer en mots - en chaînes alphabétiques - toutes les expressions non alphabétiques couramment utilisées dans les textes : les chiffres, les dates, les numéros de téléphone, les adresses mail, ...

Par exemple, il faut remplacer la chaîne "12h45" par la suite de mots "douze heures quarante cinq" ou "treize heures moins le quart". Ou encore la suite de chaînes "1 234,7 %" par la suite de mots "mille deux cent trente quatre virgule sept pour cent".

Il n'existe pas vraiment de règles générales permettant de traiter tous les cas qui peuvent être rencontrés, d'autant que de nouveaux types apparaissent constamment

comme les adresses mail ou les adresses de page web. Du coup, le formatage apparaît comme une collection plutôt disparate de traitements ad-hoc en réaménagement constant.

Actuellement le formatage ne traite que certains types de chaînes ; les nombres (entiers et fractionnaires), les fractions, les dates, les adresses web et les adresses mail. Parmi les trous les plus importants : les horaires, les numéros de téléphone (séparés par des blancs ou par des points), les grands nombres aérés par des blancs, les nombres ordinaux (“1er”, “5ème”, ...), les smiley.

Il faut noter que le formatage est la phase de traitement où il y a le plus de modifications entre LiaPhon et LliaPhon. A la fois dans les fonctionnalités - il y a plus de trous dans la version actuelle, mais certains trous de la version d’origine ont été bouchés - et dans la mise en oeuvre.

La méthode générale qui a été retenue consiste à opérer en 2 temps :

Premier temps. Segmentation de chaque chaîne en sous-chaînes homogènes de l’un des 3 types suivants : alphabétique où tous les caractères sont des lettres de l’alphabet, majuscules ou minuscules, étendues aux lettres accentuées, numérique où tous les caractères sont des chiffres, fourre-tout où aucun caractère n’est de l’un des 2 types précités.

Deuxième temps. Dans certains cas, il y a regroupement d’une suite d’items pour un traitement coordonné, soit pour réaliser une conversion, soit pour marquer les items d’une certaine façon en vue des traitements ultérieurs.

Entre ces 2 opérations se glisse le calcul des types dont on a besoin pour traiter certains cas de regroupement (adresses mail et url). Le type d’un item dit s’il s’agit d’une chaîne alphabétique, d’une chaîne numérique, d’une ponctuation de fin de phrase (’,’, ’?’, ’!’ ou ’...’), d’une ponctuation de milieu de phrase (’,’, ’:’, ’;’ ou ’-’) ou d’autre chose (mais ne contenant ni lettre, ni chiffre).

2.1 Segmentation

Chaque item dont la chaîne associée n’est pas homogène est remplacé par une liste d’items dont les chaînes associées sont homogènes. Une chaîne est homogène si tous ses caractères sont d’un seul des 3 types suivants : alphabétique, numérique, autre. Les caractères alphabétiques sont les lettres ASCII majuscules et minuscules ainsi que les lettres accentuées propres au français, majuscules et minuscules (mais on n’y a pas inclus le “ae” quoique disponible, ni le “oe” qui ne l’est pas), les caractères numériques sont les chiffres de ’0’ à ’9’ et les chaînes du 3ème type ne contiennent donc que des caractères non alphanumériques éditables (sauf le blanc).

Exemples de remplacements :

12h45 -> (12 h 45)
www.biglux.org -> (www . biglux . org)
20/1/04 -> (20 / 1 / 04)

Cette technique a l'avantage de la simplicité, mais demande quelques menues adaptations pour prendre en compte les inévitables exceptions. Le français accepte en effet 2 signes non alphabétiques comme parties intégrantes des mots : le tiret et l'apostrophe (“main-d'oeuvre”, “jusqu'aujourd'hui”). On peut y adjoindre, comme la version actuelle l'a repris de la version initiale, le point terminal utilisé pour les abréviations (“m.” pour mètre, “M.” pour monsieur, “etc.”).

Les mots où apparaît un tiret sont considérés en français comme des mots uniques, donc la séparation est à éviter si l'on veut laisser ses chances à la phase d'étiquetage qui suit. La situation est plus confuse avec l'apostrophe : “l'un” est un mot unique mais pas “qu'un”.

Puisqu'il n'existe pas vraiment de règle bien définie permettant de dire quand une chaîne où est présent (au moins) un tiret, une apostrophe ou un point est un mot unique, le mieux est d'utiliser une table des mots qui posent problème et d'aller y voir si la chaîne litigieuse y est présente ou non. C'est ce que fait LLiaPhon qui utilise un fichier spécial (intitulé `lexique/speciaux`) à cet effet et ce que ne fait pas LiaPhon qui teste la présence dans le lexique standard.

En effet de bord, les locutions qui sont présentes dans le lexique standard de LiaPhon (avec un caractère ‘_’ pour marquer le blanc de séparation) ne sont pas traitées en tant que telles dans LLiaPhon. Il est préférable de traiter une locution adverbiale du genre “au fur et à mesure” comme un mot unique lors de la phase d'étiquetage grammatical qui aura du mal à retrouver ses petits, en particulier quand elle va rechercher une étiquette possible pour le mot “fur”, mais l'absence de prise en compte des locutions ne s'est pas fait sentir sur le résultat final jusqu'ici.

La phase de segmentation produit de nouveaux items qui possèdent tous un séparateur vide sauf le dernier qui hérite celui de l'item initial. En particulier, la ponctuation est maintenant séparée des mots.

La liste précédente devient :

```
( A poursuivre une idée fixe , on ne risque pas d'aller  
  bien loin . )
```

ou encore, pour être plus précis sur les séparateurs :

```
( [A| ] [poursuivre| ] [une| ] [idée| ] [fixe| ] [,| ]  
  [on| ] [ne| ] [risque| ] [pas| ] [d'aller| ] [bien| ]  
  [loin| ] [.\|n]  
)
```

2.2 Regroupement et conversion

Alors que le traitement précédent est générique, on a ici une collection de traitements ad-hoc activés par des règles spécifiques.

Les items de la liste issue de la phase de segmentation sont testés les uns après les autres pour savoir s'il faut procéder à une opération de regroupement/conversion associant cet item et d'autres qui suivent - mais peuvent aussi précéder - qui transforme une sous-liste d'items en une autre liste d'items qui prend sa place. Les tests concernant les items portent sur le type de la chaîne et la présence ou non d'un séparateur effectif. Dans certains cas il n'y a pas de substitution, mais seulement modification des items.

Les règles mises en oeuvre dans cette phase sont pour l'instant peu nombreuses. On y a noté I_i pour l'item numéro i , c_i pour la chaîne associée et s_i pour le séparateur associé.

NBR, *NBR2*, *FR*, et *DAT* sont les fonctions qui construisent les listes d'items alphabétiques à des nombres entiers, des nombres fractionnaires, des fractions et des dates. Par exemple :

```
NBR(27) = (vingt sept)
NBR2(3 . 14) = (trois point quatorze)
FR(1 / 2) = (un demi)
DAT(22 03 04) = (vingt deux mars deux mille quatre)
```

Les notations sont quelque peu simplifiées, il faudrait en fait écrire

```
NBR2('3',',','.','14') = ('trois','point','quatorze').
```

URL et *ADR* sont les fonctions de traitement des adresses url et des adresses mail. Elles ne construisent pas de nouveaux items et fixent simplement certains attributs des items qu'elles prennent en argument.

SI c_i est de type numérique

SI c_{i+1} = “%”

remplacer ($I_i I_{i+1}$) par $NBR(c_i) + (pour\ cent)$

SI c_{i+1} = “.” ou “,” et c_{i+2} est de type numérique,

remplacer ($I_i I_{i+1} I_{i+2}$) par $NBR2(c_i, c_{i+1}, c_{i+2})$

SI c_{i+1} = “/” et c_{i+2} est de type numérique,

SI c_{i+3} = “/” et c_{i+4} est de type numérique,

remplacer ($I_i I_{i+1} I_{i+2} I_{i+3} I_{i+4}$) par $DAT(c_i, c_{i+2}, c_{i+4})$

SINON (et que $s_{i+2} \neq \emptyset$)

remplacer ($I_i I_{i+1} I_{i+2}$) par $FR(c_i, c_{i+2})$

SINON (et que $s_i \neq \emptyset$)

remplacer I_i par $NBR(c_i)$

SI c_i = “http” et $s_i = \emptyset$ et c_{i+1} = “://” et $s_{i+1} = \emptyset$

chercher le premier $j > i$ tel que $s_j \neq \emptyset$

exécuter $URL(I_i, \dots, I_j)$

SI $c_i = "@"$

chercher le premier $j < i$ tel que $s_j \neq \emptyset$

chercher le premier $k > i$ tel que $s_k \neq \emptyset$

executer $ADR(I_{j+1}, \dots, I_k)$

La fonction NBR commence par construire une chaîne de caractères par concaténation de mots séparés par des blancs, puis elle construit une liste d'items par segmentation sur le séparateur ' '. Il y en a 2 versions que l'on peut choisir à la configuration de $LLiaPhon$: la version française pour laquelle $NBR(70) = (soixante\ dix)$ et la version belge - bien plus simple à mettre en oeuvre - pour laquelle $NBR(70) = (septante)$. Les fonctions $NBR2$, FR et DAT fonctionnent sur le même principe de segmentation finale d'une chaîne construite par concaténation.

RESULTAT : On n'a plus que 2 types de chaînes : les mots (où il n'y a que des lettres et les 3 symboles spéciaux) et les chaînes caractérielles.

3 Etiquetage

OBJECTIF : Associer à chaque mot une étiquette grammaticale.

C'est ici que l'on rencontre les données fondamentales nécessaires à la synthèse :

- D'abord **le lexique** - dont on a extrait les mots spéciaux et les locutions dans la phase précédente - qui contient la plupart des mots susceptibles d'être rencontrés avec leurs diverses étiquettes grammaticales possibles (le mot "voile" peut apparaître comme verbe ou comme nom ou encore comme adjectif s'il a été désaccentué) ainsi que d'autres informations associées aux couples mot/étiquette détaillées plus loin.
- Ensuite **le catalogue** qui est la liste des étiquettes grammaticales utilisées. Dans $LiaPhon$, Frédéric Béchet utilise un catalogue assez détaillé d'une centaine d'éléments qui distingue le genre et le nombre des noms, déterminants et adjectifs et le temps, voire plus pour les verbes.

Théoriquement, l'étiquetage grammatical des mots d'une phrase demande de procéder à l'analyse syntaxique de la phrase. Mais, malgré les progrès sur la question, il est toujours difficile aujourd'hui d'obtenir de façon fiable et suffisamment rapide une telle analyse. Du coup, les systèmes de synthèse de la parole mettent plutôt en oeuvre une approche probabiliste en utilisant le fait que toutes les étiquettes possibles pour un mot donné ne sont pas équiprobables, et que tous les doublets ou triplets d'étiquettes consécutives dans une phrase ne le sont pas davantage. On dispose ainsi de données

statistiques fournissant la probabilité d'avoir tel ou tel mot pour une étiquette donnée et d'avoir tel ou tel triplet d'étiquettes consécutives et on utilise les techniques du conditionnement bayésien pour avoir à l'inverse la probabilité d'avoir telle étiquette ou telle séquence d'étiquettes pour un mot ou une séquence de mots donnés.

Classiquement, le conditionnement bayésien fait intervenir 2 variables, une variable dite explicative que l'on ne peut observer directement et une variable dite expliquée qui est celle que l'on observe. La variable explicative a une influence connue sur la variable expliquée et on cherche pour une observation donnée de la valeur prise par la variable expliquée ce que pouvait bien valoir la variable explicative. Dans le schéma bayésien, l'influence de la variable explicative sur la variable expliquée prend la forme de distributions de probabilité dites a priori, l'explication de la valeur observée sur la variable expliquée prend également la forme d'une distribution de probabilité - dite a posteriori - sur les valeurs possibles de la variable explicative. Les distributions a priori portent sur la variable expliquée sachant une valeur donnée de la variable explicative; les distributions a posteriori portent sur la variable explicative sachant une valeur donnée de la variable expliquée. On parle de conditionnement parce que, dans les deux cas, il s'agit de probabilités sur l'une des variables sous la condition que l'autre ait une certaine valeur.

Dans le cas qui nous occupe ici, la variable explicative c'est l'étiquette grammaticale dont le domaine est le catalogue, la variable expliquée c'est le mot dont le domaine est le lexique. On connaît les distributions conditionnelles des mots sachant les étiquettes et on cherche la distribution conditionnelle sur les étiquettes sachant le mot, ce qui est assez immédiat à obtenir en utilisant la règle de Bayes :

$$p(e|m) = \frac{p(m|e)p(e)}{p(m)}$$

qui a l'air un peu magique mais découle simplement de la définition des probabilités conditionnelles :

$$p(x|y) = \frac{p(x, y)}{p[y]}$$

qui fournit, appliquée à notre exemple :

$$p(e, m) = p(e|m)p(m) = p(m|e)p(e)$$

Mais la situation est un peu plus complexe. Le problème ne concerne pas seulement un mot et une étiquette mais plutôt une séquence de mots et une séquence d'étiquettes et il faut considérer, pour une phrase donnée, que la variable explicative est la séquence d'étiquettes et que la variable expliquée est la séquence de mots.

Il n'y a pas de dépendances entre mots consécutifs, ni entre un mot et les étiquettes précédentes, mais il en existe une entre chaque étiquette et les étiquettes précédentes. Du

coup, il n'est pas possible de traiter le problème des rapports entre séquences d'étiquettes et séquences de mots comme une suite de problèmes élémentaires étiquette/mot indépendants.

En fait, il faut considérer que la distribution a posteriori que l'on recherche porte plutôt sur l'ensemble de toutes les séquences possibles d'étiquettes. C'est un ensemble qui peut devenir assez grand (il contient C^n éléments si C est le nombre d'étiquettes et n la longueur de la phrase) mais il se trouve - comme c'est le cas ici - que lorsqu'on s'intéresse seulement à la séquence de probabilité maximale et non à la distribution de probabilité sur l'ensemble des séquences possibles, le problème se simplifie considérablement en utilisant la programmation dynamique directe. Dans le domaine linguistique, c'est l'algorithme dit de Viterbi - détaillé plus loin - qui met en oeuvre cette technique.

Le dépendance entre étiquettes successives peut théoriquement concerner toute la phrase, mais pratiquement on s'arrête généralement à 2. C'est à dire qu'on suppose qu'une étiquette ne dépend que des 2 étiquettes précédentes (on introduit des étiquettes fictives en début de phrase) et les probabilités conditionnelles d'une étiquette sachant le couple précédent sont décrites sous forme d'une table de triplets d'étiquettes (e_1, e_2, e_3) dotés chacun d'une probabilité d'occurrence (NOTA : la probabilité est souvent remplacée dans ces tables comme dans celle des probabilités conditionnelles mot—classe par un nombre d'occurrences - dans le corpus de textes ayant servi à construire les tables. On trouve aussi assez souvent la probabilité sous la forme d'un nombre négatif qui est le logarithme de la probabilité - ça permet de remplacer les produits de probas par des sommes sur les logarithmes).

La première phase de l'étiquetage consiste à construire une structure de données (que j'appelle un 'tagger' le mot anglais pour étiqueteur) rassemblant les données dont on a besoin : le lexique, le catalogue, les probabilités conditionnelles des mots sachant les étiquettes, la table des triplets d'étiquettes et un analyseur morphologique dont je dis quelques mots plus loin.

La deuxième phase consiste à séparer le texte en phrases (parce que la dépendance entre étiquettes n'a de sens qu'au sein d'une même phrase), à ajouter en début et en fin de phrase 2 balises (les mots $\langle s \rangle$ et $\langle /s \rangle$) qui servent de marqueurs dans la suite des traitements, puis à appeler l'étiqueteur proprement dit à qui est fournie une phrase et qui retourne une séquence d'étiquettes (en fait, il prend en entrée une liste d'items et ajoute un attribut - en fait 2 : l'étiquette, une chaîne de caractères, et son code dans le catalogue, un nombre - à chacun d'entre eux).

La troisième phase vise à essayer de réaccentuer le texte quand les accents se sont égarés quelque part, par exemple quand du texte majuscule - généralement non étiqueté - a été péremptoirement transformé en minuscule dans une étape antérieure à la synthèse de parole. La technique utilise le fait qu'à chaque couple mot/étiquette est associé dans le lexique le mot d'origine, ce que la linguistique appelle un lemme. Elle est explicitée plus loin sous le doux nom de lemmatisation - qui n'existe sans doute pas dans le lexique.

Enfin la dernière phase est un prétraitement des sigles et des noms propres. Cette phase utilise un autre analyseur morphologique dont l'objet est de proposer une autre langue que le français pour les mots non reconnus.

Un mot sur l'analyse morphologique avant de détailler le fonctionnement de l'étiqueteur. Un analyseur morphologique fonctionne en gros sur le même principe qu'un étiqueteur grammatical mais opère sur les lettres à l'intérieur d'un mot plutôt que sur les mots à l'intérieur d'une phrase. La dépendance entre lettres successives remplace la dépendance entre étiquettes successives mais il n'y a que cette dépendance, pas de variable expliquée. On retrouve les tables de triplets (de lettres cette fois-ci) affectés chacun d'une probabilité. En fait, l'analyse morphologique visant à trouver une étiquette grammaticale (dans la phase 2) ou à identifier une langue étrangère (dans la phase 4), il y a une table par étiquette dans le premier cas et une table par langue dans le second et c'est la table qui donne la plus grande probabilité sur la séquence de lettres (à partir des triplets et d'un algorithme analogue à l'algorithme de Viterbi) qui gagne.

Les fichiers utilisés :

- `lex10k.sirlex` : Le lexique.
- `lm3classe.arpa.sirlex` : Le catalogue.
- `lex10k.pmc` : Les tables de dépendances pour les mots du lexique.
- `lm3classe.arpa` : La table des triplets d'étiquettes.
- `model_morpho` : Le fichier des triplets de lettres pour l'analyseur morphologique décidant d'une catégorie grammaticale.
- `model_np` : Le fichier des triplets de lettres pour l'analyseur morphologique décidant de l'appartenance à une langue.

3.1 L'étiqueteur

La procédure d'étiquetage d'une phrase passe par 3 phases :

- La construction des tables de dépendance pour les mots de la phrase.
- Déterminer la séquence d'étiquettes la plus probable.
- Affecter à chaque mot son étiquette la plus probable.

Construction des tables de dépendance L'objectif ici est pour un mot donné, d'obtenir une table décrivant pour chaque étiquette possible pour ledit mot, la probabilité d'avoir le mot sachant qu'on a l'étiquette. Il ne s'agit pas d'une distribution de probabilité bien qu'on ait donné ce nom à la structure de donnée qui décrit ces tables (la table associée au mot m donne pour chaque étiquette e_i la probabilité conditionnelle $p(m|e_i)$ alors qu'une distribution de probabilité décrirait les probabilités conditionnelles $p(e_i|m)$. On aurait bien alors $\sum_i p(m|e_i) = 1$ alors qu'il n'y a rien de tel pour les tables)

Si le mot est présent dans le lexique, alors la table associée a été calculée lors de la construction des fichiers compilés (la commande 'make ressources' faite lors de l'installation) et a été stockée dans une des structures de données attachées à l'étiqueteur - celle de type et de nom 'pmc' pour "Probas Mots sachant Classe" (Frédéric parle de classe là ou j'utilise le terme d'étiquette).

C'est ici que le calcul de la casse est utilisé. Dans le lexique, la plupart des mots sont en minuscule, ce qui fait qu'un mot présent dans le dictionnaire, mais écrit en majuscule dans le texte traité, ne sera pas reconnu. C'est pourquoi plusieurs casses, qui dépendent de la casse initiale, sont essayées avant de conclure à la non présence du mot dans le lexique.

Si, malgré tout, le mot n'est pas reconnu ou si sa table n'a pas été stockée, une table est construite en utilisant l'analyseur morphologique. De la même façon que précédemment, si l'analyseur échoue et que le mot est en casse majuscule, on essaie en casse minuscule. Si une distribution est obtenue, on procède à un écrémage de façon à ne garder que les étiquettes qui ont une probabilité supérieure à un certain seuil (l'étiqueteur morphologique peut fournir des étiquettes ayant une probabilité d'occurrence de 10^{-10} , on a retenu un seuil à 10^{-6} , sans doute encore bien trop faible).

Si le mot en entrée est une balise, la table fournie en sortie ne comporte qu'une seule étiquette, l'étiquette ZTRM. Il en est de même si le mot en entrée n'est pas de type alphabétique ou si, pour une raison ou une autre, aucune table ne peut être définie, mais, cette fois, avec l'étiquette MOTINC.

Ainsi, la fonction qui réalise cette construction : `tagger/get_distrib` rend toujours une table de dépendances.

Voilà ce qui est fourni pour la première partie de la phrase prise en exemple (c'est à peu près ce que l'on obtient sur la sortie standard si on fixe 'niv_debug = 2' dans `tagger.c` - et qu'on recompile :-). Quelques indications pour le décodage peuvent être nécessaires : d'abord les probabilités sont fournies sous forme logarithmique, donc 0 c'est pour 1 et -6.11 c'est pour 10^{-6} , ensuite les étiquettes grammaticales ne sont pas toujours évidentes, PREPADE c'est pour une préposition du groupe (d', de, à), V3S c'est pour un verbe à la troisième personne du singulier, VA3S la même chose mais pour le verbe avoir, VPPMS un participe passé masculin singulier, etc.

```
<s>      ZTRM          0.00
```

a	PREPADE	-0.80
	PREP	-6.11
	VA3S	-0.12
	V3S	-3.70
	NMS	-4.67
	NMP	-4.86
	poursuivre VINF	-2.52
une	PINDFS	-1.20
	NFS	-3.47
	DETFS	-0.80
	CHIF	-4.26
	AFS	-4.43
idée	NFS	-2.60
fixe	VPPMS	-2.87
	V3S	-3.20
	V1S	-3.78
	AMS	-3.36
	AFS	-4.17
	ADV	-6.39
,	MOTINC	0.00

L’algorithme de Viterbi L’objectif ici est d’obtenir la séquence d’étiquettes ayant la probabilité a posteriori la plus grande.

Si l’on note $E = e_1e_2 \dots e_n$ la séquence d’étiquettes et $M = m_1m_2 \dots m_n$ la phrase, alors on note $p(E|M)$ la probabilité a posteriori de E sachant que l’on a observé M . Ce que l’on cherche, c’est parmi toutes les séquences E possibles pour M , celle qui maximise $p(E|M)$. En fait, par définition des probabilités conditionnelles, on a $p(E|M) = \frac{p(E,M)}{p(M)}$ où $p(E, M)$ est la probabilité d’occurrence du couple (E, M) parmi tous les couples possibles et $p(M)$ est la probabilité d’occurrence de M parmi toutes les phrases (de longueur n) possibles. Puisque la même valeur $p(M)$ apparaît partout, maximiser $p(E|M)$ ou maximiser $p(E, M)$ est équivalent.

On peut calculer $p(E, M)$ par récurrence en partant de la fin de la façon suivante. Si on note $E_n = E$ et $M_n = M$ et que l’on introduit E_{n-1} et M_{n-1} pour désigner les séquences E et M auxquelles on a soustrait leur dernier élément, on peut écrire : $p(E_n, M_n) = p(m_n|E, M_{n-1})p(e_n|E_{n-1}, M_{n-1})p(E_{n-1}, M_{n-1})$ (toujours par définition des

probabilités conditionnelles). En utilisant le fait que le mot m_n ne dépend que de e_n et que e_n ne dépend que des 2 étiquettes précédentes, on obtient :

$$p(E_n, M_n) = p(m_n|e_n)p(e_n|e_{n-1}, e_{n-2})p(E_{n-1}, M_{n-1})$$

C'est une relation de récurrence et on peut poursuivre, en introduisant le couple (m_{n-1}, e_{n-1}) comme on a introduit le couple (m_n, e_n) , puis, de proche en proche, jusqu'au début de la phrase.

$$p(E_n, M_n) = p(m_n|e_n)p(e_n|e_{n-1}, e_{n-2})p(m_{n-1}|e_{n-1})p(e_{n-1}|e_{n-2}, e_{n-3})p(E_{n-2}, M_{n-2})$$

Les probabilités $p(m_i, e_i)$ sont définies dans les tables de dépendances que l'on vient d'obtenir et $p(e_i|e_{i-1}, e_{i-2})$ est la probabilité conditionnelle associée au triplet (e_{i-2}, e_{i-1}, e_i) , donc on dispose de toutes les données permettant de calculer $p(E, M)$.

L'algorithme de Viterbi permet d'éviter de faire ce calcul pour toutes les séquences E_n possibles en utilisant le fait que la dépendance entre les étiquettes ne porte que sur les 2 précédentes et non sur toutes les précédentes.

Si l'on considère un couple d'étiquettes possibles (e_{i-1}, e_i) pour le couple de mots (m_{i-1}, m_i) , il est inutile de disposer des probabilités maximales pour toutes les séquences E_i finissant par le couple (e_{i-1}, e_i) (puisque seul ce couple intervient pour la suite des calculs de $i + 1$ à n et pas du tout le début de la séquence), il suffit de connaître le maximum de probabilité sur toutes ces séquences E_i et de l'attacher au couple (e_{i-1}, e_i) au rang i . Ce qui vient d'être dit est valable pour tous les rangs de la phrase et permet de considérer les séquences possibles E_n comme des chemins différents dans un même graphe.

Ce graphe est organisé en n niveaux. Les sommets au niveau i sont tous les couples d'étiquettes possibles sur les 2 mots (m_{i-1}, m_i) . Les arcs n'existent qu'entre niveaux consécutifs, un arc ne peut exister entre un état $s_1 = (e_{i-1}, e_i)$ au niveau i et un état $s_2 = (e_i^*, e_{i+1})$ au niveau $i + 1$ que si $e_i = e_i^*$ et il est alors étiqueté par e_{i+1} . Si on note P_1 la probabilité optimale pour arriver en s_1 , celle d'arriver en s_2 en passant par s_1 vaut $P_2^1 = p(m_{i+1}|e_{i+1})p(e_{i+1}|e_i, e_{i-1})P_1$ et la probabilité P_2 finalement attachée à s_2 sera la probabilité maximale pour toutes les façons d'arriver à s_2 depuis le niveau i .

Pour donner une idée de ce graphe et de son utilisation, commençons par simplifier quelque peu les tables de dépendance obtenues précédemment.

<s>	ZTRM	0.00
a	PREPADE	-0.80
	VA3S	-0.12
poursuivre	VINF	-2.52

une	PINDFS	-1.20
	DETFS	-0.80
idée	NFS	-2.60
fixe	VPPMS	-2.87
	V3S	-3.20
	AMS	-3.36
,	MOTINC	0.00

Conventionnellement, le premier niveau du graphe est le niveau 0 qui ne contient que l'état $s_0 = (\text{ZTRM}, \text{ZTRM})$.

Au niveau 1, associé au mot **a**, il y a 2 états :

$s_{1,1} = (\text{ZTRM}, \text{PREPADE})$ et $s_{1,2} = (\text{ZTRM}, \text{VA3S})$.

Au niveau 2, associé au mot **poursuivre**, les états

$s_{2,1} = (\text{PREPADE}, \text{VINF})$ et $s_{2,2} = (\text{VA3S}, \text{VINF})$.

Au niveau 3, associé au mot **une**, les états

$s_{3,1} = (\text{VINF}, \text{PINDFS})$ et $s_{3,2} = (\text{VINF}, \text{DETFS})$.

etc.

Conventionnellement, la probabilité attachée à l'état unique du niveau 0 vaut 1.

Au niveau 1, la probabilité sur l'état $s_{1,1} = (\text{ZTRM}, \text{PREPADE})$ vaut alors :

$P_{1,1} = p(\text{a} | \text{PREPADE})p(\text{PREPADE} | \text{ZTRM}, \text{ZTRM})$

avec $p(\text{a} | \text{PREPADE}) = 10^{-0.8}$ d'après la table de dépendance et

$p(\text{a} | \text{PREPADE})p(\text{PREPADE} | \text{ZTRM}, \text{ZTRM}) = 10^{-1.83}$ d'après le fichier des triplets

(à la ligne ZTRM ZTRM PREPADE).

Au niveau 2, la probabilité sur l'état $s_{2,1} = (\text{PREPADE}, \text{VINF})$ vaut

$P_{2,1} = p(\text{poursuivre} | \text{VINF})p(\text{VINF} | \text{ZTRM}, \text{PREPADE})P_{1,1}$.

Et celle sur l'état $s_{2,2} = (\text{VA3S}, \text{VINF})$ vaut

$P_{2,2} = p(\text{poursuivre} | \text{VINF})p(\text{VINF} | \text{ZTRM}, \text{VA3S})P_{1,2}$.

Au niveau 3 apparaît l'intérêt de l'algorithme. Il y a 2 façons de parvenir à l'état $s_{3,1}$, depuis $s_{2,1}$ et depuis $s_{2,2}$ (puisque chacun a comme seconde étiquette VINF, la première étiquette de $s_{3,1}$). On calcule alors $P_{3,1}^{2,1} = p(\text{une} | \text{PINDFS})p(\text{PINDFS} | \text{PREPADE}, \text{VINF})P_{2,1}$

et $P_{3,1}^{2,2} = p(\text{une} | \text{PINDFS})p(\text{PINDFS} | \text{VA3S}, \text{VINF})P_{2,2}$

et on obtient $P_{3,1} = \max(P_{3,1}^{2,1}, P_{3,1}^{2,2})$. On opère de même sur $s_{3,2}$.

Ce qui suit est la reprise aménagée de ce que fournit l'étiqueteur en mode "infos complémentaires". Chaque niveau i est identifié par la séquence de 3 mots (m_{i-2}, m_{i-1}, m_i) (en début de phrase, l'absence de mot est marquée par le symbole $_$). Au niveau i , un état $s_i = (e_{i-1}, e_i)$ est représentée par la séquence (e_{i-2}, e_{i-1}, e_i) suivie de la probabilité d'arriver en s_i depuis l'état $s_{i-1} = (e_{i-2}, e_{i-1})$. Ainsi, quand il y a plusieurs façons

d'arriver en s_i , il y a plusieurs lignes successives et la bonne probabilité à retenir pour cette état est la probabilité maximale sur les diverses lignes marquée par un signe *.

_ _ a

ZTRM ZTRM PREPADE = -2.63

ZTRM ZTRM VA3S = -1.84

_ a poursuivre

ZTRM PREPADE VINF = -7.13

ZTRM VA3S VINF = -5.55

a poursuivre une

PREPADE VINF PINDFS = -11.74

VA3S VINF PINDFS = -10.66 *

PREPADE VINF DETFS = -8.71

VA3S VINF DETFS = -7.67 *

poursuivre une idée

VINF PINDFS NFS = -13.96

VINF DETFS NFS = -10.34

une idée fixe

PINDFS NFS VPPMS = -19.75

DETFS NFS VPPMS = -16.41 *

PINDFS NFS V3S = -18.82

DETFS NFS V3S = -15.19 *

PINDFS NFS AMS = -20.23

DETFS NFS AMS = -16.98 *

idée fixe ,

NFS VPPMS MOTINC = -18.72

NFS V3S MOTINC = -17.56

NFS AMS MOTINC = -18.69

On constate au niveau 6 qu'il y a 3 états, c'est parce que la phrase n'est pas terminée. En fin de phrase, on rencontre la seconde balise d'étiquette associée **ZTRM** et il n'y a qu'un état au dernier niveau qui est le couple (**MOTINC**, **ZTRM**). La probabilité calculée sur cet état donne la probabilité maximale sur la phrase.

Affectation des étiquettes aux mots En sélectionnant en chaque état la probabilité maximale sur l'ensemble de façons d'y arriver, l'algorithme de Viterbi garde également la trace de l'état initial. Une fois trouvée la probabilité maximale sur l'état unique du dernier niveau, il est immédiat de retrouver la séquence d'états constituant le chemin optimal.

Les étiquettes associées aux arcs du chemin optimal sont alors copiées dans un attribut spécial des tables de dépendance associées à chaque mot, puis des tables vers les items.

RESULTAT A chaque item de la phrase traitée est associé un label (la chaîne de caractère décrivant l'étiquette associée à ce mot dans cette phrase) et un code numérique qui est son numéro dans le catalogue (des étiquettes grammaticales).

A noter, avec stupeur et consternation, que dans notre exemple, l'étiqueteur n'a pas été capable de déterminer correctement l'étiquette du premier mot - il prétend **V43S**, le verbe avoir à la troisième personne du singulier, alors qu'il s'agit de **PREPADE**, la proposition "à". Des erreurs de ce type sont en fait courantes (Bien qu'il soit difficile de comparer des étiqueteurs qui n'utilisent pas le même catalogue, un bon étiqueteur connaît encore couramment dans les 10% d'échecs) mais n'ont que rarement des conséquences sur la prononciation. Ainsi, dans notre exemple, le premier mot n'est pas correctement étiqueté, mais cela n'aura aucun effet sur les traitements ultérieurs (même prononciation pour "a" et "à" et aucun risque de liaison erronée puisqu'il ne peut pas y en avoir).

3.2 Lemmatisation

Le lemme associé à un couple mot/étiquette est le mot de base dont dérive le mot en question quand il porte cette étiquette : par exemple **vive** a pour lemme **vif** si c'est un adjectif, **vivre** si c'est un verbe.

Ce type d'information est disponible dans le fichier ('lex10k' ou 'lex80k' qui contient pour chaque mot une liste de triplets étiquette/proba/lemme) utilisé pour la construction

du lexique et pour celle des tables de probabilités des mots sachant les étiquettes grammaticales. Elle est perdue dans le lexique stocké en mémoire vive (qui n'est utilisé que pour savoir si un mot est connu du lexique et retourner son code) mais elle est présente dans les tables de probabilités (à chaque mot et pour chaque étiquette qu'il peut porter, on trouve la proba du mot sachant l'étiquette et le lemme d'où provient le mot quand il porte cette étiquette). Ces tables permettent donc de retourner le lemme d'un couple mot/étiquette comme elle retournent la probabilité conditionnelle.

Pour pouvoir utiliser cette information en vue de réaccentuer un mot, l'idée est d'ajouter au lexique initial ('lex10k' ou 'lex80k') pour tout mot accentué le même mot sans les accents avec la même liste de triplets étiquette/proba/lemme (et où donc les lemmes sont toujours accentués). Quand un mot nonaccentué a été doté d'une étiquette par l'étiqueteur, le lemme associé au couple mot/étiquette redonne ainsi une forme accentuée.

3.3 Prétraitement des sigles et des noms propres

Il s'agit de traitements préalables à la phonétisation qui visent à utiliser les résultats de l'étiquetage pour identifier les mots qui devront être prononcés en utilisant des règles spéciales ne correspondant pas au français standard. Ils sont réalisés pour chaque item de la phrase.

Le premier test identifie un mot comme un sigle si l'étiquette associée est **XSOC** ou encore s'il est entièrement en majuscules et que l'étiquette est **MOTINC**. Un flag associé à l'item donne le résultat du test.

Le second test identifie un mot comme un nom propre si l'étiquette associée commence par la lettre **X** (ce qui couvre les noms de villes, de pays, de société, les noms de famille et les prénoms) ou encore s'il commence par une majuscule et que son étiquette est **MOTINC**. Si le résultat est positif, le second analyseur morphologique dont il a été question précédemment décide de la langue d'appartenance; le résultat, un nombre entre 1 et 8 désignant l'une des 8 langues possibles, devient un attribut de l'item.

4 Phonétisation

OBJECTIF : Associer à chaque mot une liste de phonèmes.

Les phonèmes constituent les unités de prononciation. On peut en trouver la liste dans tout bon dictionnaire. Le Petit Robert (page xxi) dénombre ainsi 16 voyelles dont 4 nasales - spécialité française : 20 consonnes (dont le **h** aspiré et deux phonèmes empruntés à l'anglais et à l'arabo-espagnol, les 3 n'étant pas acceptés par **MBROLA** aux dernières nouvelles) et 3 semi-consonnes (que l'on trouve par exemple dans les mots *paille*, *oui* et *lui*).

Il existe un alphabet international des phonèmes qui utilise des caractères assez bizarres - qu'on ne rencontre même pas en mathématiques, c'est dire - et qui ont du être traduits sous une forme plus accessible aux ordinateurs dans des codages divers (SAMPA ou MRPA par exemple) qui utilisent un ou 2 caractères ASCII pour représenter un phonème. Cela reste assez confus pour un linguiste mal entraîné. Frédéric Béchet a introduit dans **LiaPhon** un codage systématique des phonèmes par 2 lettres qui est beaucoup plus clair, par exemple `on` au lieu de `o~` ou `uy` au lieu de `H` pour la prononciation de `lui`. C'est ce codage que nous utiliserons par la suite .

La première idée pour atteindre cet objectif est de rassembler toutes ces informations dans un lexique. C'est ce qui a été fait pour **FranFest** - la version française encore embryonnaire de **Festival** - mais ce lexique est assez lourd et ne suffit pas à traiter tous les problèmes. L'approche de **LiaPhon** est d'utiliser des règles contextuelles - transcription d'un groupe de lettres en fonction des caractéristiques de celles qui les entourent - le lexique ne subsistant que pour les mots exceptionnels dont la prononciation échappe aux règles.

D'autres règles contextuelles, prenant comme unité élémentaire non plus un groupe de lettres mais un mot, sont utilisées pour décider de la présence ou non d'une liaison et de la façon de la prononcer.

La procédure de phonétisation d'une phrase passe par 3 phases :

- La détection des liaisons.
- La transcription phonétique des mots avec, en final, une étape de prise en compte des liaisons détectées.
- Traitements spécialisés post-transcription : traitement du 'e' muet et enchaînements vocalliques.

A noter que dans **LliaPhon**, comme dans **LiaPhon**, les liaisons sont détectées avant le traitement des mots eux-mêmes et que leur effet sur la prononciation est ajoutée après. Dans **FranFest**, le traitement des liaisons, détection et effets sur la prononciation ont lieu après la transcription phonétique des mots.

Les fichiers utilisés :

- `regles_1.pro3` : Le fichier des règles concernant les liaisons.
- `h_aspi.sirlex` : Le lexique des mots commençant par un 'h' aspiré.
- `french01.pron` : Le fichier des règles 'lts' pour le français.
- `list_exep` : Le fichier des exceptions
- `desigle.pron` : Le fichier des règles indiquant si un sigle doit être lu ou épilé.

- `epeler_sig.pron` : Le fichier des règles pour les sigles épelés.
- `lire_sig.pron` : Le fichier des règles pour les sigles lus.
- `propername_1.pron` à `propername_8.pron` : Les 8 fichiers de règles pour les noms propres d'origine étrangère.
- `rule_phon.pro` : Le fichier des règles de "post-traitement".

4.1 Le traitement des liaisons

Quand un mot se termine par une consonne - souvent non prononcée mais pas toujours - et que le mot suivant dans la même phrase commence par une voyelle ou un `h` muet, il peut s'avérer souhaitable voire indispensable de prononcer la consonne finale. modifier.

La prononciation d'une consonne de liaison est en effet souvent distincte de sa prononciation dans les autres cas (par exemple `dix` se prononce `ddiiss` en fin de phrase et `ddiizz` quand il y a liaison). La liaison la plus courante (plus de 50% des cas) concerne les lettres `s,z,x` prononcées `zz` mais on trouve également `d,t,n,r,p,g` et `f`. Il y a d'autres phénomènes comme la dénasalisation des voyelles nasales (dont la graphie termine en `n`) mais une liaison ne peut survenir que sur l'une de ces consonnes en fin de mot et une voyelle ou un `h` muet en début du suivant

Quand on constate la possibilité d'une liaison, il y a encore 3 cas de figure. La liaison peut être obligatoire, interdite ou facultative.

Exemple : `Les enfants en ont assez`

La liaison entre `les` et `enfants` - comme celle entre `en` et `ont` - est obligatoire, celle entre `enfants` et `en` est interdite et celle entre `ont` et `assez` est facultative.

Le traitement des liaisons utilise un fichier de règles contextuelles et le lexique des mots commençant par un 'h' aspiré. Le contexte d'une liaison entre 2 mots est défini par une fenêtre de 4 couples mots/étiquettes autour de la liaison, 2 avant et 2 après. Divers tests portant sur ce contexte permettent de décider de l'existence ou non d'une liaison à ajouter en tête du troisième mot et d'une éventuelle modification des derniers phonèmes du second. Le lexique des mots en 'h' est indispensable pour décider s'il y a ou non liaison quand le troisième mot commence par un 'h' (interdite sur un 'h' aspiré, autorisée sur un 'h' muet).

Quelques exemples de règles :

```
context(10, <m1, <"neuf",c2>, <g3,c3>, m4>,
        "-ff", "V ",
        "neuf devant heures, hommes, ans")
->ou_bien(g3, ["heures", "hommes", "ans"]);

liaison(34, <m1, <g2,c2>, <g3,c3>, m4>,
```

```

    "||", " ",
    "substantif pluriel devant adjec, adv, verbe, auxi")
-> ou_bien_debut(c2,["NFP","NMP"])
    ou_bien_debut(c3,["A","V"]);

liaison(56, <m1, <"avant","PREP">, <g3,c3>, m4>,
    "", " T ",
    "entre avant et hier,eux,elle,elles")
->ou_bien(g3,["hier","eux","elle","elles"]);

```

Les éléments d'une règle sont, dans l'ordre :

- D'abord le mot-clé `liaison` ou `context` qui n'a pas d'incidence sur la suite. Puis un numéro de règle qui n'en a pas davantage.
- Le contexte de la liaison sous la forme d'un quadruplet de filtres (pattern in english) entre chevrons. Chaque filtre peut s'écrire sous la forme m_i ou sous la forme $\langle g_i, c_i \rangle$ quand le mot (g est là pour "graphie") ou l'étiquette (c est là pour "code grammatical") apparaissent en arguments de l'un des tests qui suivent le symbole '->. Quand on n'a pas besoin de g_i ou de c_i comme variable parce que le filtre désigne en fait une constante, celle-ci prend leur place.
- La modification éventuelle à apporter en fin du mot g_2 et la liaison éventuelle à ajouter en tête du mot g_3 . Ainsi, lorsque la règle numéro 10 s'applique, il faut retirer le phonème `ff` en fin de g_2 (qui vaut "neuf" pour que la règle s'applique) et ajouter la liaison `V` (remplacée ensuite par le phonème `vv` en tête du mot suivant. La chaîne `||` comme modification désigne en fait une interdiction.
- Un message donnant un exemple justifiant la règle.
- Après le symbole `->` (dont la justification existe mais n'a guère d'intérêt) une liste de tests qui doivent tous être vrais pour que la règle s'applique. Dans les règles prises en exemple, `ou_bien(v, [c1, c2, ...])` est vrai si et seulement si la partie du contexte désignée par `v` (mot ou étiquette) est l'un des éléments de la liste de constantes entre crochets, `ou_bien_debut(v, [c1, c2, ...])` est analogue sauf que les constantes c_i doivent constituer tout ou partie initiale de la chaîne désignée par `v` (il faut noter que dans le catalogue des étiquettes, les étiquettes apparentées commencent souvent de la même façon, aussi le test sur `c3` dans la règle 34 est vrai pour toute étiquette désignant un verbe, un adverbe ou un adjectif).

Seuls 2 tests apparaissent dans les exemples retenus mais il y en a d'autres comme `ou_bien_fin`, `different_debut`, `polysyllabique` qui demande qu'un mot comporte plusieurs syllabes ou encore `h_aspire` qui demande que le mot soit dans le lexique spécial joint aux règles de liaison.

A noter que le test polysyllabique (qui n'apparaît que dans une règle, tout comme le test parent monosyllabique) demande à ce que le mot testé soit syllabisé. Cette opération étant également utilisée - et de façon beaucoup plus intensive - pour la transcription phonétique proprement dite, elle est présentée plus loin.

4.2 La transcription phonétique

La transcription phonétique utilise essentiellement un catalogue de règles accompagné d'une liste d'exceptions. Et pour traiter les sigles et les noms propres d'origine étrangère, 11 catalogues de règles supplémentaires, heureusement beaucoup plus petits.

Toutes ces règles ont la même structure (laquelle n'est pas identique, mais plutôt plus simple que celle des règles portant sur les liaisons).

Quelques exemples de règles (extraites de 'french01.pron' le fichier des règles nominales de prononciation du français) :

```

regle(75, <" ", "ch","oeur".S." ">, "kk", "", "choeur") -> ;
regle(137, <" ", "ch","a ris">, "kk", "", "charismatique") -> ;
regle(355, <" ma ", "ch","o".S." ">, "ttch", "", "macho") -> ;
regle(971, <" psy ", "ch","a">, "kk", "", "psychanalyse") -> ;
regle(1004, <1, "ch", C>, "kk", "", "chrome") -> ;
regle(1045, <1, "ch", l1>, "ch", "", "cheval") -> ;

regle(776, <" ex prè", "s", " ">, "", "ADV", "exprès") -> ;
regle(777, <" ex prè", "s", " ">, "ss", "", "exprès") -> ;

regle(1152, <1, "a", l1>, "aa", "", "matin") -> ;
regle(1155, <1, "à", l1>, "aa", "", "là") -> ;
regle(1156, <1, "b", l1>, "bb", "", "ballon") -> ;
regle(1157, <1, "ç", l1>, "ss", "", "garçon") -> ;
regle(1158, <1, "c", l1>, "kk", "", "cri") -> ;
regle(1159, <1, "d", l1>, "dd", "", "dur") -> ;
regle(1163, <1, "é", l1>, "ei", "", "été") -> ;
regle(1164, <1, "e", l1>, "eu", "", "autrement") -> ;
regle(1165, <1, "f", l1>, "ff", "", "fille") -> ;

```

On a regroupé en tête une (petite) partie des règles qui interviennent pour décider de la prononciation de la chaîne "ch". Puis 2 règles qui permettent, en utilisant l'étiquette grammaticale du mot, de distinguer deux prononciations différentes pour la même graphie. Enfin une partie des règles les plus générales qui décident de la prononciation basique des lettres.

Les éléments d'une règle sont, dans l'ordre :

- Le mot-clé **regle** suivi d'un numéro de règle, non utilisés par les traitements.
- Un triplet de filtres entre chevrons, désignant respectivement la partie gauche, la séquence elle-même et la partie droite de la séquence de lettres dont la règle fournit la prononciation. La partie centrale est toujours une chaîne sans blancs, les parties droites et gauches sont le plus souvent des chaînes syllabisées (où les syllabes sont séparées par des blancs et où le début ou la fin de mot est indiquée par 2 blancs successifs - l'algorithme de syllabisation, dont il a été déjà question à propos des liaisons, est présenté plus loin) mais elles peuvent aussi être des concaténations de chaînes syllabisées et de symboles codant pour un type de lettre (voyelle, consonne, occlusive ou liquide) ou cherchant à couvrir les formes singulières et plurielles du même mot, ou encore masculine et féminine (par exemple " ".C.C1."ent " codant pour un blanc suivi de 2 consonnes puis de la chaîne "ent" ou " nel le".S codant pour le couple de syllabes "nel", "le" suivi éventuellement de la lettre 's').
- La prononciation associée à la partie centrale du triplet précédent.
- Une chaîne décrivant une contrainte sur l'étiquette grammaticale. L'exemple de la règle 776 demande que le mot en cours de traitement soit un adverbe. On trouve également des contraintes demandant que le mot soit un verbe ou, au contraire, n'importe quoi sauf un verbe.
- Un exemple justifiant le rôle de la règle.
- Le symbole -> est encore plus inutile que pour les règles contextuelles des liaisons, il n'y a jamais rien derrière, sauf ';' le marqueur de fin de règle.

Les règles de transcription sont plus simples que celles de liaison mais l'ordre dans lequel elles sont testées est plus compliqué.

Pour les liaisons, la première règle qui s'applique - dans l'ordre du tableau en mémoire qui est le même que dans le fichier et qui n'a pas forcément à voir avec la numérotation des règles - est appliquée et on passe au mot suivant.

Pour la transcription, il y a souvent plusieurs règles qui s'appliquent sur un segment de mot donné et il faut d'abord tester les plus spécifiques avant d'essayer les plus générales. Ensuite, il faut d'abord essayer les règles qui acceptent le plus grand segment de mot possible. Par exemple, la règle 971 (la chaîne ch dans psychanalyse) est plus spécifique que la règle 1045 (la chaîne ch en général) et, en outre il faut essayer les règles portant sur la chaîne "ch" avant celles portant sur "c". L'ordre dans lequel les règles sont testées dépend donc à la fois de leur spécificité plus ou moins grande - qui est fixée une fois pour toutes et calculée à partir d'indicateurs de complexité des règles et qu'on retrouve en gros dans la numérotation - et du mot en cours de traitement qui impose un ordre supplémentaire non déterminable une fois pour toutes.

La transcription traite les mots les uns après les autres. Ses diverses phases, dans l'ordre, sont les suivantes :

1. Est-ce une exception ? Les exceptions sont reconnues par examen du lexique spécial associé aux règles nominales de transcription. Ce lexique est une liste de couples mots/transcriptions phonétiques et lorsque le mot en cours de traitement y est présent, l'obtention de sa transcription est immédiate.

L'exception est recherchée sous sa forme native, puis, s'il y a échec, en passant à une casse sans majuscules, et finalement en ôtant l'éventuel 's' final.

Trouver une règle spécifique pour chaque exception ne serait pas forcément bien compliqué, mais il faut s'assurer que ladite règle ne se déclenche que pour cette exception et cela, vu les conditions de déclenchement des règles, c'est beaucoup moins simple. Du coup, il est beaucoup plus simple de garder des exceptions tant que le coût d'exploration du lexique ne devient pas exorbitant.

2. Est-ce un sigle ? Si c'est le cas (la détection ayant eu lieu en fin de la phase d'étiquetage) sont utilisées les 3 catalogues de règles dédiées au traitement de sigle. Le premier catalogue répond à la question : le sigle doit-il être lu, comme c'est le cas pour ATTAC ou le CULTE, ou épilé, comme c'est le cas pour OMC ou SNCF ? Ensuite selon la réponse est utilisé un des 2 autres.
3. Est-ce un nom propre étranger ? Là aussi, la détection de la langue a eu lieu en fin d'étiquetage et elle fournit le numéro du catalogue de règles de prononciation dédiées à cette langue.
4. A noter une proposition de traitement d'un cas d'homographie hétérophone particulièrement retors : celui opposant les fils de famille et les fils du discours.
5. Si ce n'est rien de tout cela, sont alors utilisées les règles nominales.
6. En final, si la transcription a été faite, sont prises en compte les éventuelles modifications de fin de mot et ajouts de liaison en tête de mot déterminées dans l'étape précédente de traitement des liaisons.

Et si tout a échoué, ce qui ne peut survenir que sur une chaîne ou tous les caractères sont non alphabétiques, sont utilisées, éventuellement, des règles spéciales de prononciation des signes de ponctuation.

4.3 Post-Traitements

Il reste un dernier catalogue de règles, provenant du fichier `rule_phon.pro` beaucoup plus réduit que le fichier des règles nominales, pour réaliser quelques menues opérations cosmétiques sur les transcriptions phonétiques. Il s'agit essentiellement du traitement du

'e' muet et de la transformation de certains phonèmes dans les enchainements vocalliques (par exemple la transformation de *ii* en *yy* dans "il y est").

Ces règles font souvent intervenir plusieurs mots (3 dans l'exemple précédent) et sur un mode distinct des règles de liaison. Quoiqu'elles portent directement sur les structures phonétiques et non plus sur les mots eux-mêmes, elles ont en fait la même structure que les règles nominales. Comme elles peuvent porter sur plusieurs mots, le contexte phonétique sur lequel travaillent les règles est constitué par concaténation des transcriptions du mot précédent, du mot courant et du mot suivant.

Il faut noter une indécision assez marquée en ce qui concerne le traitement du 'e' muet puisqu'une grande partie des règles portant sur la question de savoir quand prononcer le 'e' muet étaient transformées en commentaires dans le fichier original. Et l'une des rares qui restaient a été supprimée fin 2003 parce qu'il en restait encore trop à notre goût.

4.4 La syllabisation

Objectif : segmenter un mot en syllabes.

D'après le Petit Robert, une syllabe est "une unité phonétique fondamentale, groupe de consonnes et de voyelles qui se prononcent d'une seule émission de voix". Pas vraiment opératoire pour définir un algorithme. Heureusement le même Petit Robert définit la syllabation comme suit : "Répartition d'un système d'articulations en syllabes, soit opérée spontanément par le sujet parlant, soit reconnue par le phonéticien d'après la définition qu'il adopte de la syllabe" (d'après un certain Maruzeau). On apprend ainsi du même coup qu'on dit "syllabation" et non "syllabisation" et que chaque phonéticien a sa propre définition de ce qu'est une syllabe.

Frédéric Béchet, plutôt qu'une définition, fournit 3 listes de lettres liées qui doivent forcément apparaître ensemble dans une syllabe : des couples de voyelles, des couples de consonnes et des triplets de consonnes (exemples : au, oi, gr, ch, chr et sch) et un ensemble de règles de création de coupures entre syllabes selon que l'on rencontre 2 voyelles successives, une voyelle suivie d'une, de deux, de trois ou de plus de 4 consonnes (exemple : si l'on rencontre 3 consonnes derrière une voyelle et que le triplet est lié, la coupure a lieu avant le triplet, sinon si les 2 dernières sont liées, la coupure a lieu entre la première et la seconde consonne, sinon entre la seconde et la troisième). Contrairement aux règles de traitement des liaisons et de la phonétisation, celles-ci, avec les inévitables exceptions permettant de traiter le cas de la voyelle 'y' à qui il arrive de se comporter comme une consonne et les couples 'qu', 'gu' et 'cu' qui doivent aussi être traités comme des consonnes uniques, sont codées en dur et non traduites sous forme de données.

Parmi les exceptions, il faut également noter le traitement spécial du trait d'union.

5 Prosodie

OBJECTIF : Associer à chaque phonème une durée et construire le contour mélodique.

Tout vient à point à qui est arrivé jusque là, mais ça peu encore prendre un peu de temps.

6 Annexe : Les traitements linguistiques

Une synthèse vocale vise à traduire un document écrit sous forme parlée. Il faut généralement disposer d'un système de pré-traitement qui traduit le document initial - un mail, une page html ou pire, une copie d'écran - sous forme d'un texte stocké sous un format informatique standard (ASCII, maintenant Unicode) et d'un système de génération du signal sonore à partir d'une transcription phonétique du texte. Le passage du texte à sa transcription phonétique constitue le coeur de la synthèse.

Le texte est une suite de lettres (dont un certain nombre accentuées en français), de chiffres et de signes divers (ponctuation, parenthèses et signaux cabalistiques genre arobas ou perluette)

La transcription phonétique est une suite de phonèmes (sons élémentaires) chacun doté d'une durée et le tout enrobé dans un contour mélodique décrivant l'évolution de la fréquence fondamentale du signal au cours du temps.

6.1 Phonétisation

C'est le noyau minimal de la synthèse, l'opération qui réalise la transcription orthographique phonétique.

Une première idée pour réaliser une telle transcription est de définir un lexique associant à chaque mot susceptible d'être rencontré dans le texte sa transcription phonétique.

Une entrée dans un tel lexique serait par exemple :

poursuivre (pp ou rr ss uy vv rr)

On utilise ici la notation proposée par Frédéric Béchet pour les phonèmes, bien plus pratique que les standards courants.

Remarque 1 C'est plutôt lourd. Les mots doivent être présents dans le lexique de transcription sous leurs diverses formes (genre et nombre pour les adjectifs, conjugaison pour les verbes), ce qui peut amener à un lexique de 100 000 à 200 000 entrées. Ce qui implique taille mémoire nécessaire et temps d'accès à l'information un peu trop grands.

Remarque 2 C'est loin d'être complet.

- Il manquera inmanquablement tous les mots en création constante : les sigles, les noms propres, les emprunts aux autres langues.
- Les liaisons entre mots successifs (indispensables en français, essayer de prononcer "Les enfants en ont assez" sans les liaisons) ne peuvent pas être traitées par un tel lexique seul.

Pour remédier à ces inconvénients, la deuxième idée consiste à définir des règles de transcription lettres phonèmes capables de s'adapter à tous les mots, y compris des mots

inventés de toutes pièces. Dans la littérature sur la question, anglophone comme il se doit, ces règles sont dites **règles lts** (pour “letter to sounds”).

Une règle lts comporte généralement un contexte gauche, le groupe de lettres dont il s’agit d’obtenir la transcription, et un contexte droit. Les contextes gauche et droit peuvent être des groupes de caractères ou des filtres (“pattern” in english) acceptant certains types de groupes (par exemple * qui filtre pour n’importe quel groupe, y compris vide)

Exemples :

(*,g,a) -> gg pour dire que la lettre ‘g’ n’importe où dans le mot mais suivie de la lettre ‘a’ se prononce gg

(*,g,i) -> jj pour dire que ce n’est plus le cas si ‘g’ est suivie de ‘i’.

L’ordre d’activation des règles est capital. Il faut prendre compte à activer les règles spécifiques, comme (*,g,i) -> jj avant les règles générales, comme (*,g*,*) -> gg.

Un catalogue raisonnable de telles règles pour le français comporte un millier de règles. C’est le cas dans LiaPhon qui définit en outre des catalogues annexes pour les sigles et pour les noms propres d’origine étrangère.

Cela n’étonnera que ceux qui ont oublié leur séjour à l’école primaire et leur apprentissage de la grammaire avec ses perpétuelles exceptions et exceptions aux exceptions, mais il ne semble pas y avoir en linguistique de règles sans exceptions. Aussi, au **catalogue de règles** est toujours adjoint un **lexique des exceptions** qui rassemble tous les mots dont la transcription phonétique ne découle pas des règles. C’est en fait simplement le lexique initial dont on a retiré les mots dont la transcription correcte est justement couverte par les règles.

6.2 Etiquetage grammatical

Etiqueter un mot, c’est décider de sa catégorie grammaticale (verbe, nom, adjectif, adverbe, préposition, ...). Les catégories peuvent être plus ou moins fines, LiaPhon utilise un catalogue qui compte 104 étiquettes.

Il y a plusieurs raisons de procéder à l’étiquetage des mots :

- L’existence d’une liaison entre 2 mots dépend de leur orthographe - le premier doit finir sur une consonne, le second commencer sur une voyelle - mais aussi très souvent de leurs catégories grammaticales respectives. Par exemple, selon que l’adjectif et avant ou après le mot, la liaison peut être obligatoire ou interdite : “un excellent argument” demande une liaison mais “un argument excellent” l’interdit.
- La prosodie fait intervenir des unités tonales, des groupes de mots, dont les frontières sont souvent liés à des changements importants de catégorie grammaticales : par exemple entre un nom ou un verbe et un article ou une conjonction qui le suit.

- Enfin il y a les fameux **homographes hétérophones** qui, comme le nom l’indique, s’écrivent de façon identique mais se prononcent différemment. Par exemple “Le président et son sous-fifre président la réunion”, “Les portions que nous portions sont tombées” ou encore “Il est parti vers l’est”. La catégorie grammaticale permet généralement de lever l’ambiguïté - nom contre verbe dans les 3 exemples - mais ce n’est pas toujours le cas comme dans “Ce sont ses fils qui ont coupés les fils”.

Comment procède-t’on ? Normalement, il faudrait procéder à l’analyse syntaxique de la phrase pour identifier correctement les catégories grammaticales des mots. Mais une telle analyse est, encore aujourd’hui malgré des légions de chercheurs en linguistique super-compétents, hors de portée.

Du coup, on procède, littéralement, un peu au hasard en cherchant à recopier ce qui se passe, en moyenne, dans les textes couramment rencontrés. En effet, quand un mot peut posséder plusieurs étiquettes, celles-ci ne sont généralement pas présentes avec la même fréquence. De la même façon, les étiquettes ou couples ou triplets d’étiquettes ne sont pas non plus présents avec la même fréquence.

Et si l’on dispose de statistiques décrivant ces diverses fréquences dans un corpus (un ensemble de textes de même type et style : journalistique, scientifique, juridique, ...) donné, on peut avoir une bonne chance d’obtenir les bonnes étiquettes pour une phrase pas trop éloignée du style du corpus en utilisant les techniques de l’inférence statistique.

Plus précisément, pour ce faire, il faut disposer des probabilités d’occurrence de tous les couples possibles mot/étiquette et des probabilités d’occurrence de tous les triplets d’étiquettes (dans la pratique courante, on s’arrête aux triplets qui semblent donner satisfaction). Puis on utilise le conditionnement bayésien pour obtenir la séquence d’étiquettes de probabilité maximale sachant la séquence de mots.

On ne dispose pas forcément de statistiques sur tous les mots, non seulement les catégories de mots en création continue comme les sigles, mais aussi les formes fléchies (conjugaisons des verbes) peu usitées comme l’anecdotique imparfait du subjonctif qui ne sert plus qu’à faire des calembours. On utilise alors un analyseur morphologique qui utilise les mêmes principes généraux que l’analyseur syntaxique, mais cette fois-ci au niveau des lettres et non plus au niveau des mots (par exemple un mot un peu grand qui se termine par “ment” est presque assurément un adverbe).

L’étiquetage grammatical étant indispensable au traitement des liaisons, il doit avoir lieu avant la phonétisation.

6.3 Formattage

En entrée de l’étiquetage grammatical, il faut disposer de mots, c’est-à-dire des suites de lettres. Or, on trouve souvent dans les textes des suites de chiffres voire des suites composites de chiffres, de lettres et de caractères divers comme la barre de fraction ou la perluette.

Le formattage est l'opération - préalable à l'étiquetage donc - qui sépare les tokens lexicaux (présents dans les tables de l'étiqueteur et acceptables par les règles de phonétisation) des autres qui doivent, soit être traités de façon particulière comme ponctuation, soit être ignorés comme les lignes de tirets utilisés pour marquer une séparation, soit traduits sous forme d'un ou de plusieurs tokens lexicaux.

Les tokens lexicaux sont formés des lettres de l'alphabet, plus, en français, d'un certain nombre de lettres accentuées, et de 2 signes particuliers : l'apostrophe et le tiret (exemple : c'est-à-dire) qui peuvent être considérés comme membres à part entière de l'alphabet.

Dans LiaPhon, Frédéric Béchet ajoute le point qui apparaît essentiellement en fin de mot pour marquer les abbréviations et dont la présence dans l'alphabet est plus discutable.

Les tokens non-lexicaux qui doivent être traduits sont, entre autres :

- Les nombres. Il faut remplacer le token "1995" par la suite "mille neuf cent quatre vingt quinze", ou "mille neuf cent nonante cinq" pour la version belge.
- Les dates. "le 1/8/03" doit devenir "le premier août deux mille trois".
- Les fractions. "1/3" se traduit par "un tiers" et "1/5" par "un cinquième".
- Les adresses email. "abul@cnam.fr" devient la suite "ABUL chez CNAM point FR" où les mots sont mis en majuscules pour être traités comme des sigles - lus pour les 2 premiers, épilé pour le dernier.
- Les horaires, les numéros de téléphone, les sommes d'argent, les abbréviations, etc.

Quant aux autres tokens non-lexicaux, il n'est pas toujours évident de décider de leur traitement. Ainsi, le point marque la fin de phrase mais peut aussi être présent pour marquer une abbréviations. Le tiret est une lettre de l'alphabet, mais c'est aussi un caractère de ponctuation et un marqueur de changement de locuteur dans les dialogues. Et le mode d'énonciation sur les structures enchassées : passages entre guillemets, entre parenthèses, entre chevrons, crochets ou accolades n'obéit pas toujours à des règles simples et figées une fois pour toutes (par exemple, il faut éviter de traduire "((" par "ouvrir la parenthèse" quand on lit du Lisp).

6.4 Calcul de la prosodie

La prosodie, c'est la musique de la phrase. Les phonèmes sont des sons qui possèdent un timbre comme les instruments de musique : une certaine répartition de l'énergie sur une fréquence fondamentale et ses harmoniques, répartition qui évolue au cours du temps. Le profil de répartition de l'énergie sur l'échelle des fréquences est typique du phonème

et est particulièrement bien marquée et constante pour les voyelles, alors que l'évolution de cette répartition est particulièrement rapide sur les consonnes.

Ce profil de l'intensité sonore sur l'échelle des fréquences peut facilement - pour un locuteur donné - être stocké dans une table de phonèmes. Mais comme il varie beaucoup en début et en fin de phonèmes, il s'est avéré indispensable de stocker la variation plutôt que la partie stable - souvent très brève - sous forme d'une table de diphtonges (couples de phonèmes consécutifs) pour avoir une reproduction correcte des suites de phonèmes. C'est le principe de base des logiciels - du type MBROLA - de rendu sonore des transcriptions phonétiques : au lieu de rendre une suite de phonèmes, l'algorithme construit un signal sonore à partir d'une suite de diphtonges entre lesquels il procède à une interpolation assez élémentaire - mais néanmoins brevetée sans aucun complexe en ce qui concerne MBROLA.

Quand on dispose d'une table de diphtonges, il ne reste plus qu'à fixer la durée de chaque phonème et l'évolution de la fréquence fondamentale au cours du temps. C'est cela qui constitue le calcul de la prosodie.

Tout autant que le vocabulaire et la grammaire, la prosodie est typique d'une langue. Ainsi, l'anglais marque certaines voyelles des mots - et il faut savoir lesquelles - et accepte de transformer les autres en bouillie phonétique alors que le français marque la fin de certains groupes de phonèmes - et ce n'est pas immédiat d'identifier lesquels - mais n'accepte d'éliminer que le 'e' muet et encore pas toujours.

Pour le français, la notion de base est celle d'unité tonale, une suite de phonèmes commençant et finissant sur une frontière de mot, pas trop longue et qui obéit souvent à des contraintes syntaxiques. Ainsi, une frontière d'unité tonale apparaît le plus souvent entre un mot dont l'étiquette grammaticale est un nom, un adjectif ou un verbe et un autre dont l'étiquette est une préposition, une conjonction ou un déterminant.

Les unités tonales sont regroupées en unités d'intonation dont les frontières sont généralement bien marquées par la ponctuation.

Des règles définissant la durée des phonèmes (augmentation sur les voyelles de fin d'unité tonale par exemple) et l'évolution de la fréquence fondamentale (un peu difficile à préciser dans une annexe) sont disponibles dans la littérature scientifique sur la question, mais très peu cohérentes entre elles et rarement clairement explicitées et définitives.

Dans **LiaPhon**, le calcul de prosodie mis en oeuvre n'est pas fameux et est annoncé comme tel. Et **LLiaPhon** ne fait pas mieux. Dans **FranFest** on ne dispose que d'un calcul de prosodie adapté à l'anglais et les résultats appliqués au français sont franchement pénibles.

Ainsi donc - oyez, oyez, braves gens - nous ne disposons dans le domaine des synthèses libres de rien de bien sérieux sur la question. C'est le manque le plus criant.

References

- [1] Frédéric Béchet. Lia-Phon : un système complet de phonétisation de textes. LIA, Université d'Avignon.