# Rapport de stage de L3 : Reconnaissances de syllabes et trancription automatique du chant du canari

Maxime Cauté

Août 2019*

## Introduction

Le langage est un formidable outil de communication employé au quotidien par de nombreux agents humains et non-humains. Pourtant, notre compréhension de son traitement (apprentissage, compréhension...) reste à ce jour parfois limitée, notamment dans le cadre des langues humaines.

Moins complexes, les chants aviens sont d'excellents objets d'étude afin d'approfondir nos connaissances dans ce domaine. En effet, leur syntaxe repose sur des "chunks", regroupements de briques élémentaires, assemblées pour former le chant. De plus, l'absence de sémantique élude les difficultés qui y sont liées. Cette organisation est particulièrement remarquable car elle est fortement partagée avec les langages humains : les phonèmes forment des syllabes, qui forment des mots, qui eux-mêmes forment les phrases. La compréhension des processus associés est donc une question fondamentale pour les neurosciences.

L'objectif de ce stage fut donc d'étudier une méthode de reconnaissance et de transcription automatique du chant des canaris, qui a la particularité d'abriter une syntaxe particulièrement riche. Ceci permettrait non seulement d'améliorer notre compréhension de ceux-ci, mais également de développer de nouveaux outils utilisables pour des expériences neuroscientifiques. L'essentiel du temps fut consacré au développement d'une méthode de reconnaissance automatique des syllabes dans les chants d'oiseaux.

*retravaillé en avril 2020

# 1 État de l'art

## 1.1 Historique de l'étude des chants

Darwin [**Dar882**], déjà, en 1882, notait les similarités entre l'apprentissage des vocalisations chez l'homme et du chant chez l'oiseau. Cela déclencha par la suite plusieurs études comparatives entre les deux, aux niveaux cognitif, génomique ou comportemental. En 1961, Thorpe écrit sur les mécanismes biologiques de la communication vocale chez l'oiseau [**Tho61**]. En 1985, Konishi se penche sur le rapport entre le comportemental et le neuronal dans les chants d'oiseaux [**Kon85**]. En 2011, leur syntaxe est étudiée formellement par Berwick et asso. [**Ber11**].

Le cas spécifique du canari est analysé en 2013 par Markovitz et asso. [**Mar13**]. Leur travail souligne l'existence de chaînes de Markov dans les chants de canari. Cependant, la théorie d'un comportement markovien est contrariée par l'ordre élevé de ces chaînes. Celles-ci soulignent la nécessité d'une forme de mémoire, assimilable celle dite de travail, chez le canari. Un enjeu majeur de ces études est de savoir si les oiseaux peuvent utiliser des procédés récursifs. C'est dans ce contexte que la reconnaisse automatique fut envisagée.

## 1.2 Reconnaissance du chant des canaris

La reconnaissance automatique de structure au sein des chants étant particulièrement intéressante pour traiter de larges quantités de données expérimentales, plusieurs telles tentatives ont déjà été réalisées, avec diverses méthodes.

En 1996, Anderson et asso. adoptèrent une Machine à Support Vectoriel ($MSV$) couplée à la Déformation Temporelle Dynamique ($DTD$) pour annoter le chant du diamant mandarin [**And96**]. En 2012, Tan et asso. [**Tan12**] classifient à l'aide de DTD. En 2016, Kaetwip et asso. [**Kae16**] emploient la DTD sur l'espèce des viréos de Cassin, en priorisant les régions proéminentes du signal.

# 2 Contexte scientifique

## 2.1 Le chant du canari

Le canari est un oiseau domestique capable de produire une grande variété de sons. Son chant est organisé en syllabes dont la répétition forme des phrases. Ces phrases se suivent dans un ordre précis, selon des schémas probabilistes. Certains enchaînements de phrases, très fréquents, forment ce que l'on appelle des "chunks".

On observe dans l'enchaînement de ces chunks une structure en chaînes de Markov d'ordre relativement élevé. Markovitz [**Mar13**] identifie des chaînes d'ordre supérieur à 4. En d'autres termes, le choix d'une syllabe chantée dépend de ce qui a pu être chanté 4 phrases plus tôt, voire plus. Cette complexité est donc l'un des intérêts majeurs du canari comme sujet d'étude. Cela nécessite en effet l'emploi d'une forme de "mémoire de travail", sujet d'étude important en neurosciences.

On peut ajouter à cela une grande richesse du lexique chez cet oiseau, qui contient plusieurs dizaines de syllabes, variant d'un individu à l'autre. Il est de plus remarquable que ce lexique varie d'année en année, contrairement à d'autres espèces comme le diamant mandarin où il se fixe une fois l'âge adulte atteint.

## 2.2 Base de données

Notre banque de travail est un ensemble de 450 chants enregistrés par l'équipe de Catherine Del Negro à Orsay. Ceux-ci ont été émis par différents canaris, reconnus par des bagues colorées.

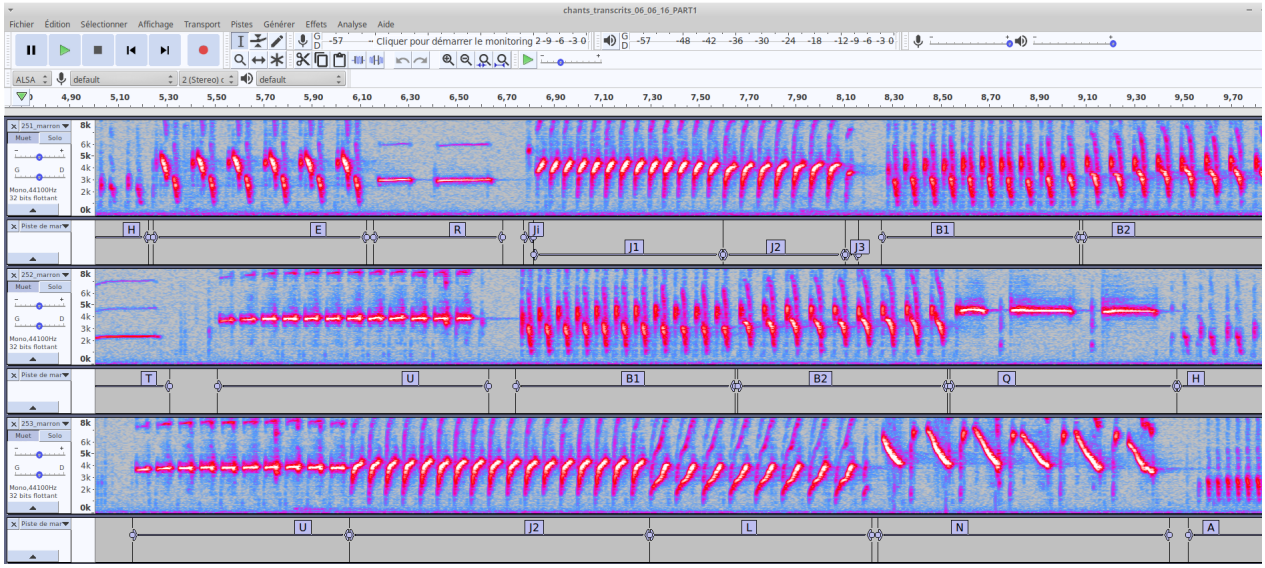



FIGURE 1 – Spectrogrammes de chants

Ces chants durent entre 7”36 et 46”52. Les paramètres d'enregistrements ont pu varier entre deux prises. Ces chants ont été annotés au niveau de leurs phrases dans leur totalité. Il est cependant à noter que quelques erreurs ont été par la suite repérées.

Un extrait des spectrogrammes des chants, visualisés par le logiciel *Audacity*, est montré en figure 1. Les marqueurs délimitent les phrases, dont les portions périodiquement répétées sont les syllabes.

# 3 Contribution

Afin d'analyser les chants des canaris, il est nécessaire de les identifier, en annotant les phrases selon le type de syllabe répétée. Ce travail a déjà été en partie réalisé à la main, mais il est souhaité, devant la grande quantité de chants, d'automatiser cette tâche

Pour cela, il faut bien sûr analyser différentes portions du signal pour être capable de les identifier comme une composante éventuelle du chant (phonème, syllabe, phrase). Cependant, ces portions doivent alors être préalablement isolées, ce qui ajoute un problème supplémentaire.

Le processus d'annotation automatique se décline alors comme suit :

1. recueil des données du chants ;
2. pré-traitement éventuel ;
3. sélection des portions à identifier ;
4. identification de ces portions ;
5. post-traitement éventuel ;
6. sortie de la syntaxe trouvée.

## 3.1 Identification par DTD

La méthode d'identification s'appuie sur des mécanismes classiques de comparaison à un modèle. Cependant, une grande diversité au sein des éléments des chants considérés. En effet, tout comme pour la parole, le chant des oiseaux présente des phrases, des syllabes et des phonèmes

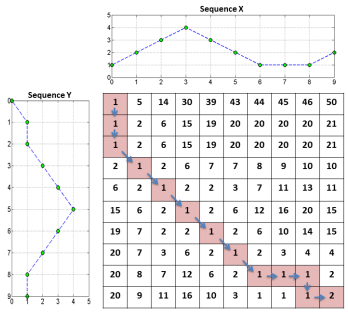



FIGURE 2 – Exemple de DTD avec mémorisation du chemin

dont la durée peut varier d'un chant à l'autre. La forme temporelle des phonèmes peut également légèrement varier.

Ainsi, la méthode de comparaison retenue est celle de la déformation temporelle dynamique (*DTD*), aussi appelée *Dynamic Time Warping* **[SakChi78]**.

Cette méthode sert à comparer deux séquences $x$ et $y$ qui peuvent ne pas être alignées dans le temps. Le principe est alors de trouver une déformation de la requête qui permet de faire plus coïncider les deux. Le coût associé à celle-ci prend donc en compte deux paramètres : la distance en valeur entre les trames reliées, et leur distance temporelle.

Or cette déformation doit bien évidemment conserver l'ordre des trames. La déformation retenue pour la trame $x_{i+1}$ depend donc de celle pour la trame $x_i$, et les coûts associés de même.

Afin de calculer efficacemment cela, nous avons donc recours à la programmation dynamique. En suivant une déformation euclidienne, le coût $DTW(i,j)$ de la sous séquence $x_0x_1...x_i$ à $y_0y_1...y_j$ est le minimum des trois valeurs suivantes.

1. $DTW(i,j-1) + C(i,j)$
2. $DTW(i-1,j-1) + C(i,j)$
3. $DTW(i-1,j) + C(i,j)$

($C$ est une fonction de coût, par exemple $C : i,j \longrightarrow ||x_i - y_j||$)

Le calcul de la distance totale par DTD revient donc à calculer $DTW(|x|-1,|y|-1)$, sous la forme d'une matrice remplie par programmation dynamique. Il nous reste donc à choisir de bons paramètres pour optimiser le résultat.

**Choix de la contrainte locale** À chaque étape, le choix de la formule suivie (dans notre exemple 1, 2 ou 3) constitue dans cette matrice le choix d'un chemin. Cette contrainte, appelée contrainte de chemin ou contrainte locale, peut varier selon les propriétées attendues de la DTD. Nous avons de notre côté pris le parti de choisir le chemin en trident**[Tan12]**, qui se définit comme suit (une illustration se trouve en figure 4).

Si $i$ et $j$ sont reliés, alors il est possible d'avoir les liaisons

1. $(i+2, j+1)$
2. $(i+1, j+1)$
3. $(i+1, j+2)$

Ce choix permet de limiter la déformation afin d'éviter des assimilations de signaux trop importants.

**Data :** Une séquence de référence $y$,
s une séquence de requête $x$
Une fonction de distance $d$
**Résultat :** Une distance entre $x$ et $y$

1 on initialise $DTW$, matrice de taille $(|x|+1, |y|+1)$, avec toutes ses valeurs à $+\infty$
2 $DTW[0,0] \leftarrow 0$
3 **pour** $1 \leq i \leq |x|$ **faire**
4  **pour** $1 \leq j \leq |y|$ **faire**
5  $cout \leftarrow d(x_i, y_j)$
6  DTW[i,j] = cout + min $\begin{vmatrix} DTW[i-1,j] & // \text{ insertion} \\ DTW[i,j-1] & // \text{ suppression} \\ DTW[i-1,j-1] & // \text{ correspondance} \end{vmatrix}$
7  **fin**
8 **fin**
9 renvoyer $DTW(|x|,|y|)$

**Algorithme 1 :** Algorithme général pour la DTD

**Choix de la contrainte globale** De même, il est possible de définir une fenêtre de chemin, aussi appelée contrainte globale. Cette fenêtre empêche les liaisons entre certains points, par exemple jugés *a priori* trop distants. Ci-après se trouvent trois contraintes classiques, illustrées en annexes par la figure 5.

— La première, appelée bande de Sakoe-Chiba, consiste en une bande centrée sur la diagonale. Elle permet ainsi une déviation limitée vis-à-vis de celle-ci (qui correspond à l'absence de déformation). Mathématiquement, pour un rayon $r$, cela revient à dire que pour une association entre les trames d'indices $i$ et $j$, il est nécessaire d'avoir $i-r \leq j \leq i+r$. Dans notre cas, face au fait que nos signaux puissent fortement varier en taille, nous avons décidé de moduler $r$ par rapport à la taille de la requête, ce qui donne, pour autoriser un flottement à 10%, $r = \frac{10}{100} \times |x|$.

— La seconde, appelée le parallélogramme d'Itakura, consiste en un parallélogramme défini par sa demi-largeur maximale. Celle ci est alors appliquée au milieu du signal. Les points initiaux et finaux ont eux une largeur de 0. Ceci permet de favoriser les déformations au centre du signal, sans déformer les extrémités.

— La troisième consiste à entraîner l'algorithme à employer une contrainte globale optimale.

Pour les étapes de reconnaissance, nous avons choisi d'employer la bande de Sakoe-Chiba, avec une largeur de 10%. Les déformations étant réparties sur toute la syllabe, le parallélogramme d'Itakura apparaissait inadapté. La fenêtre apprise fut également laissé de côté car trop complexe algorithmiquement pour notre exploration du problème.

**Passage au calcul de score** Afin d'effectuer des comparaisons, il est plus aisé de manipuler un score borné. Pour cela, nous employons une fonction de score plutôt qu'une fonction de coût dans le calcul dynamique. C'est alors le score maximal qui est retenu et non la valeur minimale. Il reste enfin à s'assurer que ces valeurs restent dans l'intervalle.

Formellement, cela devient donc (avec la contrainte de chemin en trident) :

$$DTW(i,j) = min \left| \begin{array}{l} DTW(i-1,j-2) + \frac{1}{N}[\frac{1}{2}\Theta(i,j-1) + \frac{1}{2}\Theta(i,j)] \\ DTW(i-1,j-1) + \frac{1}{N}\Theta(i,j) \\ DTW(i-2,j-1) + \frac{1}{N}[\Theta(i-1,j) + \Theta(i,j)]) \end{array} \right. \quad (1)$$

(où $N$ est le nombre de trames de la requête, et $\Theta$ une fonction de score entre $-1$ et 1)

Cela donne donc un score final entre -1 et 1. Le facteur $\frac{1}{2}$ permet justement d'avoir un facteur total pour la trame $i$ de valeur 1.

**Choix de la fonction de score** Notre choix de fonction de score s'est porté sur la similarité cosine. Celle-ci est en effet un classique adapté à la manipulation de vecteurs, et se définit comme suit :

$$CS(u,v) = \frac{< u, v >}{||u|| \times ||v||} \quad (2)$$

Il est enfin à noter que la DTD est globalement incapable de mettre en relation deux répétitions de la même période si ces répétitions sont de tailles différentes. Ainsi, l'identification ne pouvait se faire que par rapport aux phonèmes ou aux syllabes, mais pas à l'échelon phrase. Nous avons alors choisi de considérer les syllabes pour cela, car le traitement des phonèmes eut nécessité d'avoir une banque de données axée sur ceux-ci. Toutefois, il eut été envisageable, avec une telle base de donnée, de faire de la reconnaissance de phonèmes et ensuite d'appliquer un algorithme de reconnaissance des motifs pour revenir aux syllabes.

## 3.2 Sélection des requêtes

### 3.2.1 Méthode retenue : fenêtre glissante

Afin de déterminer quelles portions de signal seront soumises à la comparaison, nous avons pris le parti d'appliquer une fenêtre de reconnaissance glissante sur le signal. Le principe consiste en une fenêtre de taille fixée, que l'on décale peu à peu, en effectuant une tentative d'identification à chaque étape. Les bornes temporelles choisies pouvant englober un morceau silencieux du chant, il est donc toutefois nécessaire d'en créer un modèle.

Les fenêtres que nous avons utilisées ont toutes un décalage de 50%. Leurs tailles font 50, 150, et 300 millisecondes.

### 3.2.2 Méthode alternative : segmentation des phonèmes

Il est cependant possible d'envisager d'autres méthodes afin de poser les repères pour l'identification. Parmi elles, la segmentation automatique développée par Härmä [**Har03**] en 2003.

**Segmentation de Härmä** Le principe de cette segmentation est d'assimiler les pics d'intensité dans le spectrogramme à la marque d'un phonème individuel. Partant de ce pic, il suffit alors d'atteindre ce qui en serait les extrémités pour délimiter le phonème. Pour cela, il suffit de regarder quand l'amplitude courante devient trop faible par rapport au pic. Cela se fait chez Härmä à l'aide d'un seuil global $T$, appelé critère d'arrêt. Celui-ci intervient localement afin de délimiter les pics : lorsque l'amplitude courante est inférieure à l'amplitude du pic moins $T$. Il joue également un rôle global, en cessant la recherche lorsque l'amplitude des pics est en dessous

de l'amplitude du premier pic moins $T$. Cela permet notamment d'éviter la prise en compte de pics "fantômes" liés au bruit.

Formellement, l'algorithme se présente comme suit.

1. On calcule le spectrogramme $S(f,t)$ du signal. Ce spectrogramme est une matrice d'abscisse les fréquences $f$ et d'ordonnées les temps $t$.
2. On calcule l'amplitude maximale du spectrogramme $a_{max} = 20\log_{10}(max(|S|))$. On en déduit le seuil global $a_{max} - T$.
3. On répète en boucle les étapes 4-7 suivantes tant que l'amplitude des $a_0$ pics reste au-delà du seuil global.
4. On calcule $(f_0, t_0) = argmax(|S|)$. Ces valeurs représentent le pic d'amplitude d'un phonème.
5. On calcule l'amplitude du pic, $a_0 = 20\log_{10}(|S(f_0,t_0)|)$. On en déduit le seuil local $a_0 - T$.
6. On calcule l'intervalle $I = [t_d, ..., t_0, ..., t_f]$ où les amplitudes restent supérieures au seuil local. On parcourt donc, dans un sens puis l'autre, depuis $t_0$, les temps $t_c$ tant que, pour l'amplitude courante $a_c = 20\log_{10}(max_{f\in F}(|S(f,t_c)|)$, $a_c > a_0 - t$.
7. On redéfinit $S(I,F) = 0$ , afin de ne pas sélectionner de nouveau cette section.

Nous avons repris et adapté le code MatLab implémenté par Lindemuth en 2010.

**Segmentation par intensité** Un méthode similaire consiste à déterminer les syllabes à l'aide de l'intensité du signal même. À cette fin, le signal est mis au carré, puis est lissé par une moyenne glissante. Enfin, un seuil d'intensité permet de considérer tous les intervalles de temps où l'intensité est supérieure à ce seuil comme étant des phonèmes.

Malheureusement, ces deux méthodes se sont révélées insuffisantes. L'intensité des phonèmes varie trop fortement, et ceux-ci sont parfois trop rapprochés. Aussi, nous balancions entre sur- et sous-segmentation selon les portions du signal. De plus, les variations entre les chants accentuaient ces difficultés.

### 3.3 Pré-traitement du signal

Afin d'optimiser la reconnaissance par le DTW nous avons utilisé le descripteur des coefficients cepstraux des fréquences de Mel du signal. Leur calcul se fait par la méthode suivante, implémentée par la librairie Python *Librosa*.

1. Afin de corriger les erreurs dues à la discrétisation du signal, nous appliquons un fenêtrage de Hamming sur le signal. Ce fenêtrage permet de compenser le fait que les traitements du signal (par exemple les transformées de Fourier) s'effectuent de manière discrète et finie, quand le signal réel est continu et infini. Plus précisément, il s'agit notamment d'atténuer la coupure nette aux bords de la fenêtre d'enregistrement en y diminuant l'amplitude du signal. Mathématiquement, la fenêtre de Hamming consiste à multiplier le signal par la fonction $h$ suivante, sur l'intervalle $[0,T]$ de la fenêtre.

$$h(t) = \begin{cases} 0,56 - 0.46 \times cos(2\pi\frac{t}{T}) & \text{si } t \in [0,T] \\ 0 & \text{sinon} \end{cases} \tag{3}$$

2. Le spectrogramme du signal est calculé à l'aide d'une transformée de Fourier rapide. Le calcul est réalisé de sorte à avoir au moins 2 trames par phonème, c'est à dire 2 toutes les 300 millisecondes. Seules les fréquences pertinentes, entre 1500 et 8000 Hz,

sont conservées. Le spectrograme est ensuite passé à l'échelle de Mel. Cette échelle de fréquences consiste en la conversion des hertz par la formule suivante, où $m$ représente la valeur en mels et $f$ celle en hertz.

$$m = 1127 \times ln(1 + \frac{f}{700})$$

3. Ce spectrogramme est ensuite transformé en un cepstre de fréquences de Mel. Pour cela, on calcule un banc de filtre triangulaire. Ces filtres sont des fonctions des fréquences dans $[0; 1]$ dont la représentation est triangulaire. On applique alors un tel filtre $F_m$ au spectrogramme. On calcule ainsi la somme pondérée de tous ses coefficients : pour un spectrogramme $S$, de domaines fréquentiel $F$ et temporel $T$.

$$E_m = \sum_{\substack{t \in T \\ f \in F}} F_m(f) \times S(f, t)$$

On obtient la valeur de $E_m$, appelée énergie en sous bande.
Il reste à calculer les coefficients cepstraux à partir de ces énergies par une tranformée cosine discrète. Le k-ième est calculé par la formule suivante, où $M$ est le nombre de filtres initialement employés.

$$X_k = \sum_{m=0}^{M-1} ln(E_m) \times cos(\frac{\pi}{M}(m + \frac{1}{2})k)$$

Nous avons choisi de conserver les 24 premiers, en évinçant de plus le premier ($k = 0$), qui est lié à des fréquences très faibles, non pertinentes pour nous. De plus, nous avons employé les dérivées première et seconde de ces coefficients comme descripteurs complémentaires. Pour le calcul, ces dérivées ont le même traitement qu'un coefficient d'ordre 0.

### 3.4 Création de la banque de modèles

Il est toutefois nécessaire de constituer une banque de modèles pour effectuer les comparaisons. Pour cela, nous avons annoté les syllabes de 7 chants pour servir de référence. Ces chants subissent alors le même prétraitement que les chants automatiquement traités. Les portions de signal annotés sont ensuite rassemblées comme modèle dans une première banque d'entraînement. Notre méthode de segmentation pouvant sélectionner des morceaux de silence dans le chant, il est de plus nécessaire d'en avoir un modèle, directement récupéré sur les chants.

Toutefois, pour faciliter la généralisation et éviter un temps de traitement trop important, un seconde banque est créée afin de résumer la première. Ainsi, au sein de chaque classe, nous souhaitons obtenir un modèle qui est suffisamment proche de tous les autres. Cependant, il peut se trouver une certaine variabilité au sein d'une même classe de syllabes. La première étape est donc de regrouper ensemble les syllabes proches. La méthode retenue consiste simplement à sélectionner une syllabe comme racine de la grappe, et lui associer celles de score supérieur à $0,8$.

Il est alors possible de résumer chaque grappe à l'aide de la méthode de Moyennage Barycentrique Dynamique **[Pet10]** (*MBD*).

**Moyennage Barycentrique Dynamique** Le MBD, développé par Petitjean en 2011, vise à moyenner, en barycentres, plusieurs séquences par rapport à la DTD. Plus précisément, le MDB calcule progressivement la séquence dont la somme des distances aux autres séquences, mises au carré, est minimale.

Ces raffinements progressifs du résumé ont lieu suivant la méthode suivante.

1. La matrice de DTD est calculée entre toutes les séquences et la séquence résumée courante.
2. Cette matrice est utilisée afin de trouver les associations par DTD entre les trames de ces séquences et de la séquence résumée. Cela permet, pour chaque coordonnée de la séquence résumée, d'avoir un ensemble de coordonnées qui lui sont associées.
3. Les nouvelles coordonnées de la séquence résumée sont chacune calculée comme le barycentre des coordonnées associée du jeu de séquences original.

**Data :** Un jeu de séquences $S = \{s_1, s_2, ..., s_n\}$ à moyenner
Une séquence $m_k$, moyenne de $S$ à la $k$-ième étape
**Résultat :** Une moyenne $m_{k+1}$ pour $S$

$assocTable \leftarrow [\emptyset, \emptyset, ..., \emptyset]$ ; On initialise *assocTable*, tableau de taille $|m_k|$, qui fait correspondre à chaque point de $m_k$ les points des séquences de $S$ associés.
**pour** $s_i \in S$ **faire**
  $M, P \leftarrow DTW(s_i, m_k)$ ; On calcule les matrice de coût et de chemin par DTD entre $s_i$ et $m_k$.
  $i \leftarrow |m_k|$
  $j \leftarrow |s_i|$
  **faire**
    $assocTable[i] \leftarrow assocTable[i] \cup s_i[j]$
    $(i, j) \leftarrow P[i, j]$
  **fin tant que** $i \geq 1, j \geq 1$;
**fin**
**pour** $1 \leq i \leq |m_k|$ **faire**
  $m_{k+1} \leftarrow barycentre(assoTable[i])$
**fin**
renvoyer $m_{k+1}$

**Algorithme 2 :** MDB

Ces 3 étapes peuvent ensuite être répétées pour obtenir une approximation de plus en plus efficace d'une moyenne des séquences. Pour des raisons de complexité algorithmique, nous avons limité ces répétitions au nombre de 4.

Il est à noter que cette méthode est particulièrement intéressante pour nous par son usage de la DTD.

### 3.5 Post-traitement des données

La procédure employée jusqu'ici nous calcule, pour chaque type de syllabe, un score associé à chaque fenêtre, dénommé score temporel ci-après. Il est possible d'en observer un exemple sur la figure 3. Le lecteur y observer les courbes des scores associés à chaque syllabe, lissés sur 10 points afin d'adoucir les variations. Si certains score se démarquent par moment, on notera que ceux-ci sont à plusieurs moments caractérisés par de fortes fluctuations, rendant une identification compliquée.

Il nous faut alors utiliser ce score temporel afin de reconstruire la syntaxe de la phrase. La méthode naïve consiste en la sélection, à chaque instant, du candidat au score le plus élevé.

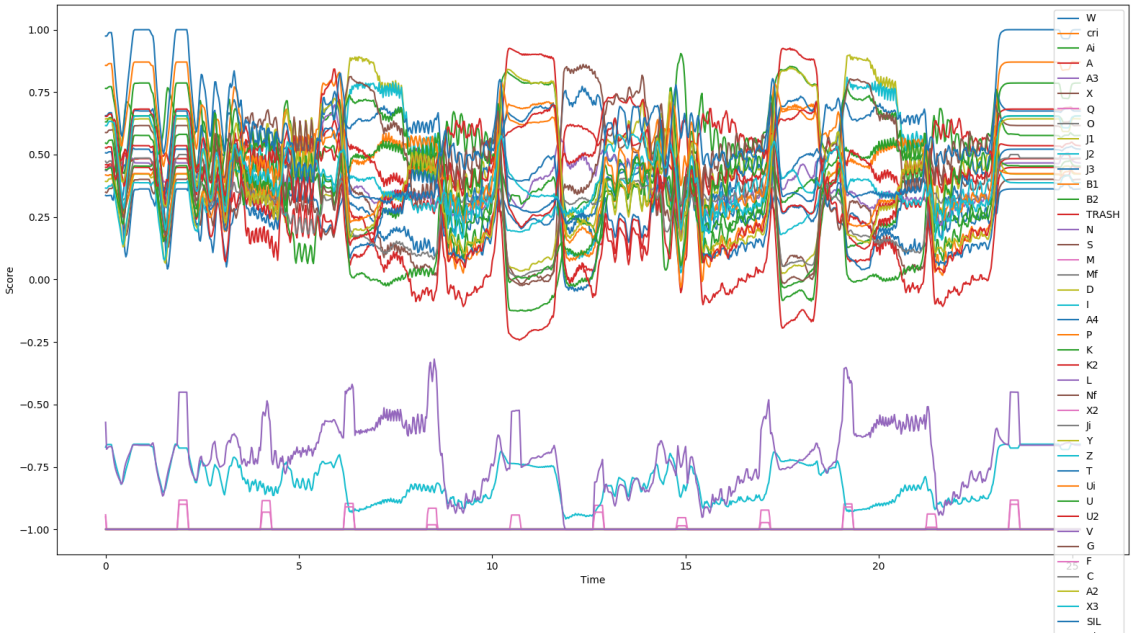



FIGURE 3 – Exemple de scores temporels, lissé sur 10 points

Cependant, les syllabes étant de durée variable, une seule fenêtre ne permet pas d'identifier efficacement toutes celles-ci. Les scores temporels sont donc calculés pour différentes tailles de fenêtre et de décalages.

Ces scores sont par la suite accordés. Pour cela, il est constitué une liste comprenant tous les marqueurs générés par ces scores. Ces marqueurs comprenent les dates de début et de fin de la syllabe potentielle, son type ainsi que son score. Si deux marqueurs sont en conflit, (c.à.d. que leurs dates se chevauchent), celui-ci est résolu en donnant la primauté au marqueur de plus grand score. Cela est appliqué jusqu'à la fin des conflits. Les marqueurs de silence sont ensuite évincés afin de ne garder que les véritables syllabes.

Un fichier de transcription est ainsi obtenu pour ce chant. La reconstruction des phrases consiste alors en un regroupement des syllabes de même type. Il est cependant possible de prévenir les erreurs par d'autres méthodes de post-traitement. Il est par exemple envisageable de marquer comme douteuses les syllabes dont le score reste trop faible. L'utilisateur peut alors être sollicité. Des méthodes statistiques peuvent également être considérées à partir des chants servant à l'entraînement de l'algorithme.

# 4 Résultats et discussion

La méthode décrite ci-avant permet donc d'obtenir une séquence de phrases automatiquement extraite des chants. Il nous reste toutefois à évaluer les performances de l'algorithme présenté.

## 4.1 Évaluation finale

L'évalutation finale de la syntaxe délivrée consiste en une comparaison des séquences obtenues. La méthode retenue fut donc le calcul de la distance d'édition entre les deux. Nous avons donc considéré la distance de Levenshtein. Sur notre jeu de chants, avec les paramètres choisis,

la distance obtenue est de 10427. En moyenne, il y a donc une distance de $\frac{10427}{54}$ soit d'environ 193 par chant. Cette distance est trop élevée, pour des chants qui ne comprennent qu'au plus quelques dizaines de phrases ! Le résultat obtenu est donc fortement erronné, et il ne nous apparaît pas suffisant pour réellement simplifier la tâche d'annotation à cause de ce large taux d'erreur. La distance espérée aurait été inférieure à la moitié de la taille d'un chant, car cela signifie globalement que la moitié des phrases était correctement identifiée.

## 4.2 Évaluation de la banque résumée

Il est important, afin d'obtenir des résultats optimaux, d'assurer que notre banque de modèles résumée soit bien fidèle à l'originale. Nous avons pour cela la matrice de similarité croisée entre les types de modèle. Cette matrice se décline en similarité maximale, moyenne et la variance dans la comparaison de deux classes. L'objectif est alors d'avoir une matrice présentant une diagonale de 1 (syllabes parfaitements reconnues) et le reste de -1 (syllabes considérées comme opposées). Les matrices sont disponibles en annexe, aux figures 6, 7 et 8. Il est possible d'y observer une diagonale marquée, malgré certains îlots de confusion. La reconnaissance individuelle des syllabes semble donc efficace.

## 4.3 Discussion

Il semblerait donc que notre méthode ne parvienne pas assez finement à analyser le chant des canaris. Si plusieurs tentatives fructueuses d'étiquettage de chants d'oiseaux ont déjà eu lieu, celles-ci ont été réalisées sur des espèces différentes du canari, aux chants moins complexes.

Malgré cela, les traitements réalisés sur le chant semblent en bonne voie pour parvenir à nos objectifs. Sur les graphiques des scores temporels, les longues phrases et les silences ont été visuellement facilement identifiables. Les syllabes plus ponctuellement présentes grimpent également en score à l'approche de leur position réelle dans le chant. Il me semble donc que la partie à perfectionner se situe au niveau du post-traitement sur les scores temporels. Les trois points suivants me paraissent ainsi être des cibles pertinentes pour l'amélioration du code.

**Post-traitement** Il pourrait être judicieux d'ajouter, en plus de la moyenne glissante, une autre composante du traitement plus explicitement axée sur l'évolution des scores de chaque syllabe. Autrement, une meilleure paramétrisation de cette moyenne pourrait parvenir à améliorer significativement les résultats. Cependant, celle-ci présente actuellement des déficiences dans deux directions opposées. Il y a tout d'abord un problème de sur-moyennage, c'est à dire que les syllabes courtes (qui ont peu de points sur la courbe) voient leur score moyen tiré par le bas, car la fenêtre de moyennage est trop grande par rapport à la syllabe. Il se trouve également un problème de sous-moyennage, c'est à dire que certaines syllabes ont un score très fluctuant (à cause de similarité avec une portion de la syllabe réelle), et que celui-ci dépasse donc parfois ponctuellement celui de la syllabe réelle. Pour pallier cela, il pourrait également être envisagé de plus varier les fenêtres glissantes de reconnaissance, voire même de les pondérer. Les autres paramètres pourraient également être optimisés.

**Optimisation des paramètres** En revanche, cette optimisation de paramètres peut s'avérer délicate au vu de la coût temporel d'une exécution. Sur les 54 chants d'environ 20 secondes utilisés pour les tests, la durée moyenne de traitement (de la lecture du signal à la création des graphiques de scores temporels) est de 463,32 secondes par chant ! Pour cette raison, nous n'avons pas exécuté une optimisation automatique, par exemple à l'aide du module python *Hyperopt*, notamment à cause du temps limité. Néanmoins, celle-ci pourrait s'avérer très bénéfique, notamment dans le cadre du choix du pré-traitement.

**Descripteurs du chant** En dehors de cela, il est également possible de chercher à améliorer la discrimination des syllabes, en s'inspirant de ce qui se fait sur les vocalisations humaines. Par exmple, Millet et asso. [**Mil19**] ont employé en juin dernier des postériogrammes (*posteriorgrams*) comme descripteurs pour la discrimination automatique de syllabes humaines. La fonction distance associée est la divergence de Kullback-Leibler. Il est également possible d'envisager une autre structure de MFCCs, par exemple en pondérant différemment les dérivées dans le descripteur, de sorte à privilégier une dérivée en particulier.

# 5 Conclusion

Ce stage aura permis le développement d'un logiciel d'extraction automatique de la syntaxe d'un chant d'oiseau.

Malgré des résultats intermédiaires encourageants, nous ne sommes pas parvenus à étiqueter automatiquement un chant de canari avec une précision suffisante pour être exploitable. Cependant, nous avons exploré un large pan des méthodes de reconnaissance. Les résultats intermédiaires obtenus peuvent ainsi nous orienter sur les méthodes à employer afin de parvenir à un algorithme particulièrement fonctionnel.

De plus, le code peut servir de base à un logiciel plus performant, car celui-ci implémente différentes structures de contrôle et de correction.

# Référence et remerciements

Ce rapport est la conclusion d'un stage de 9 semaines, dans le cadre de la formation L3 de l'ENS Rennes, réalisé sous l'encadrement de M. Xavier Hinaut dans l'équipe MNEMOSYNE du centre INRIA Bordeaux Sud-Ouest.

Je tiens à remercier M. Frédéric Alexandre et toute son équipe pour leur accueil. Je tiens à remercier tout particulièrement M. Xavier Hinaut pour le temps passé à m'encadrer et à relire ce rapport. Je remercie de plus M. Nicolas Rougier pour ses conseils et nos échanges, toujours instructifs. Je tiens également à remercier MM. Anthony Strock et Alexis Juven pour leur aide précieuse durant ce stage. Enfin, je remercie à nouveau toute l'équipe, pour leur accueil, leur gentillesse quotidienne, et l'aide qu'ils ont pu m'apporter au cours de cette période en leur compagnie.

Je remercie également très sincèrement M. Roman Ursu pour le code de la segmentation par intensité, ses propositions avisées, et le temps qu'il aura passé à me présenter l'IMN et ses travaux.

Merci enfin à tout le personnel de l'IMN de Bordeaux pour leur accueil, et les sympathiques échanges que nous avons eus.

# Annexes

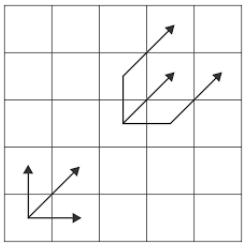
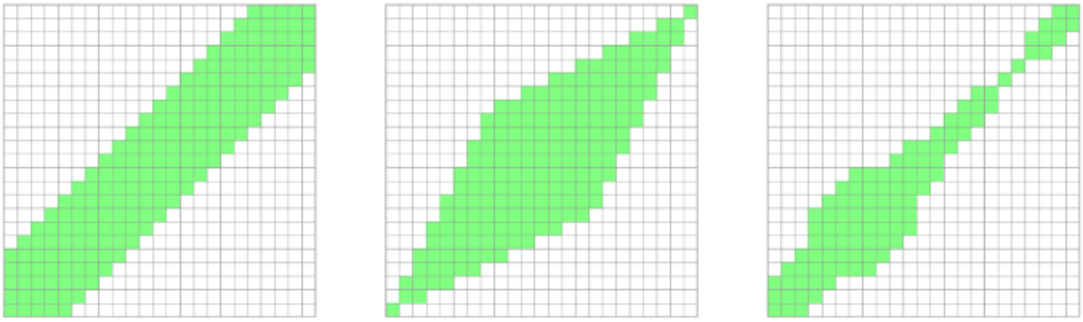


FIGURE 4 – Contraintes locales euclidienne et en trident



(a) Sakoe-Chiba band (b) Itakura parallelogram (c) learned window

FIGURE 5 – Exemples de fenêtres de chemin

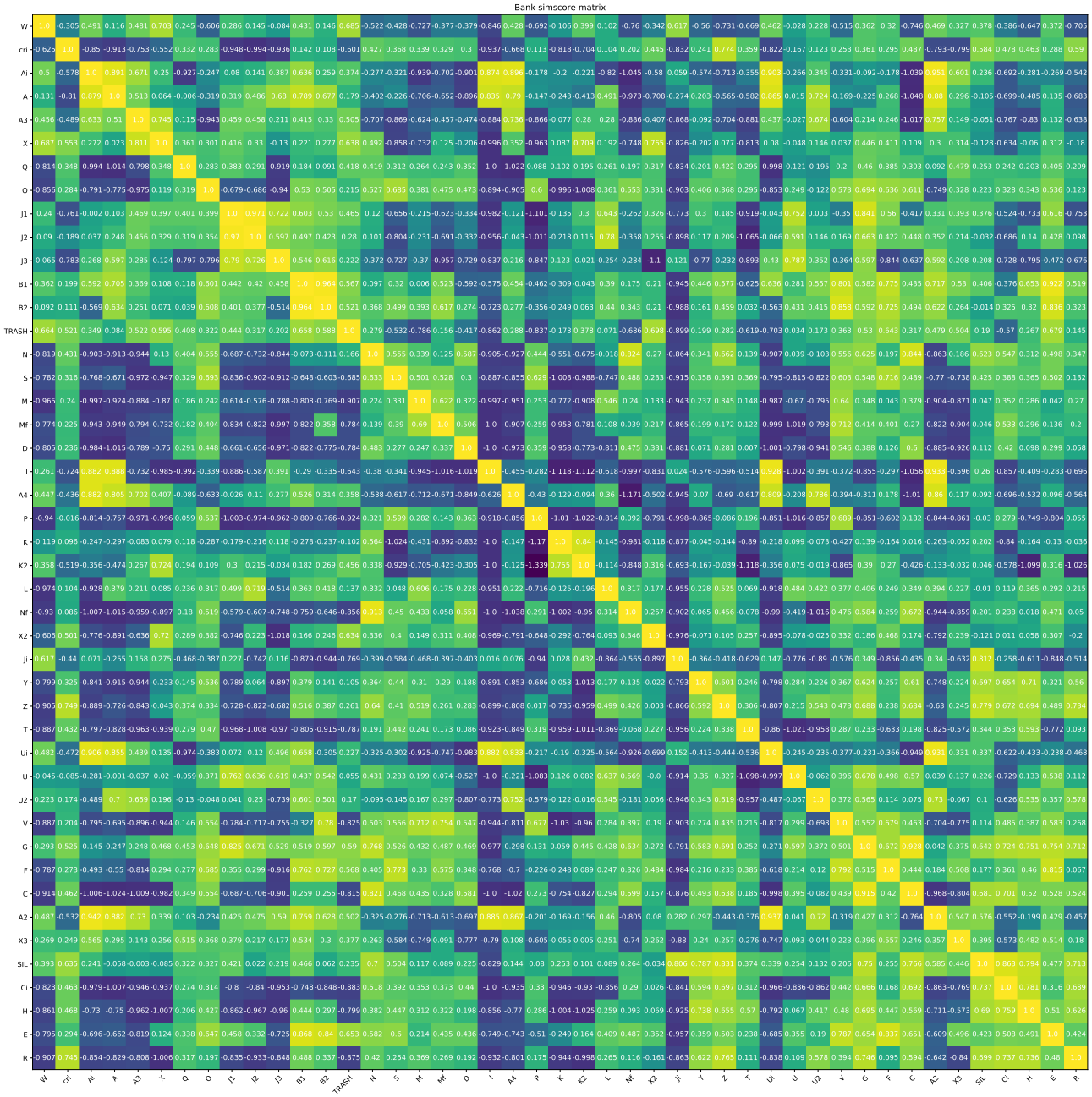



FIGURE 6 – Matrice de similarité maximale pour la banque résumée

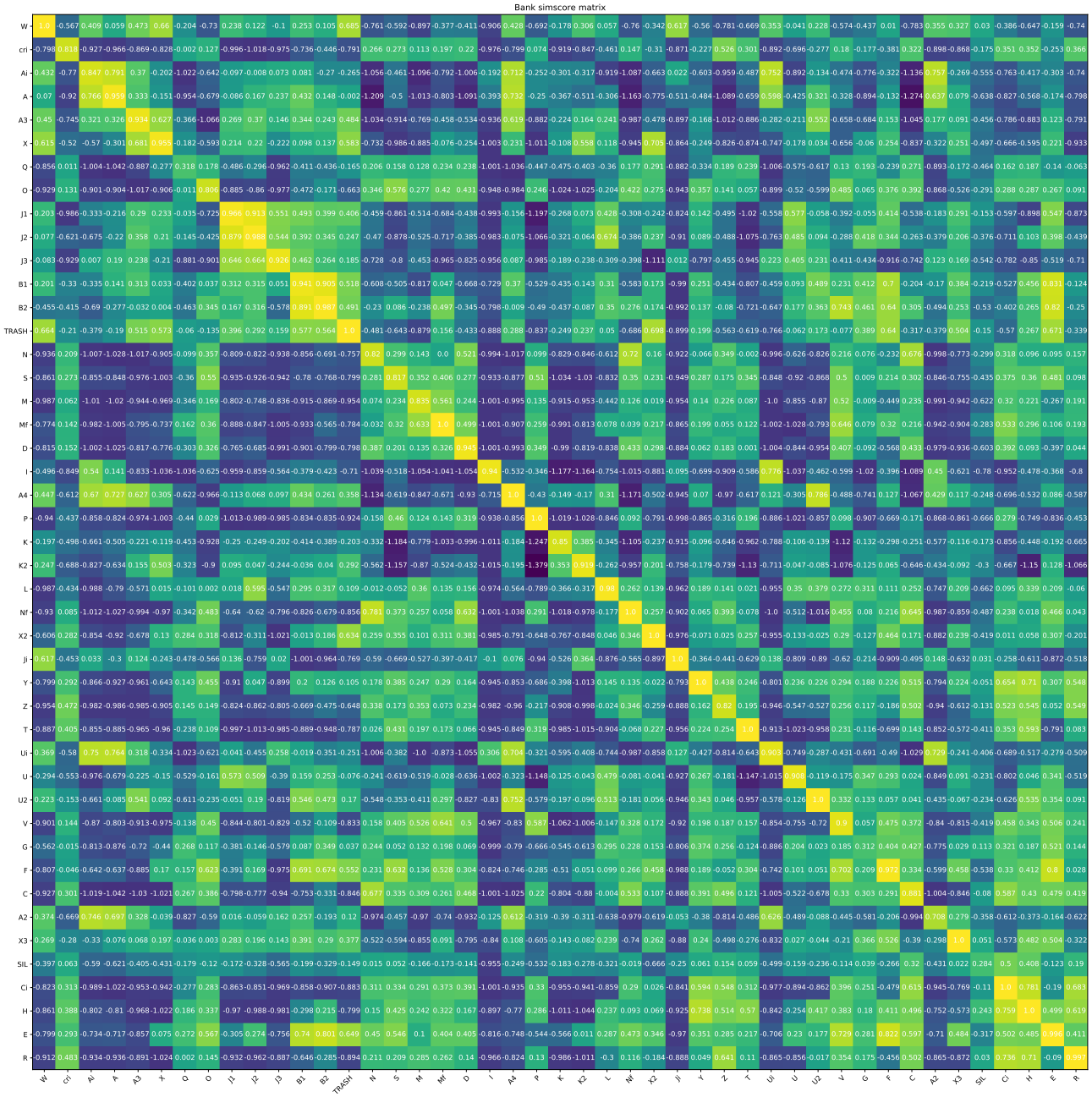



FIGURE 7 – Matrice de similarité moyenne pour la banque résumée

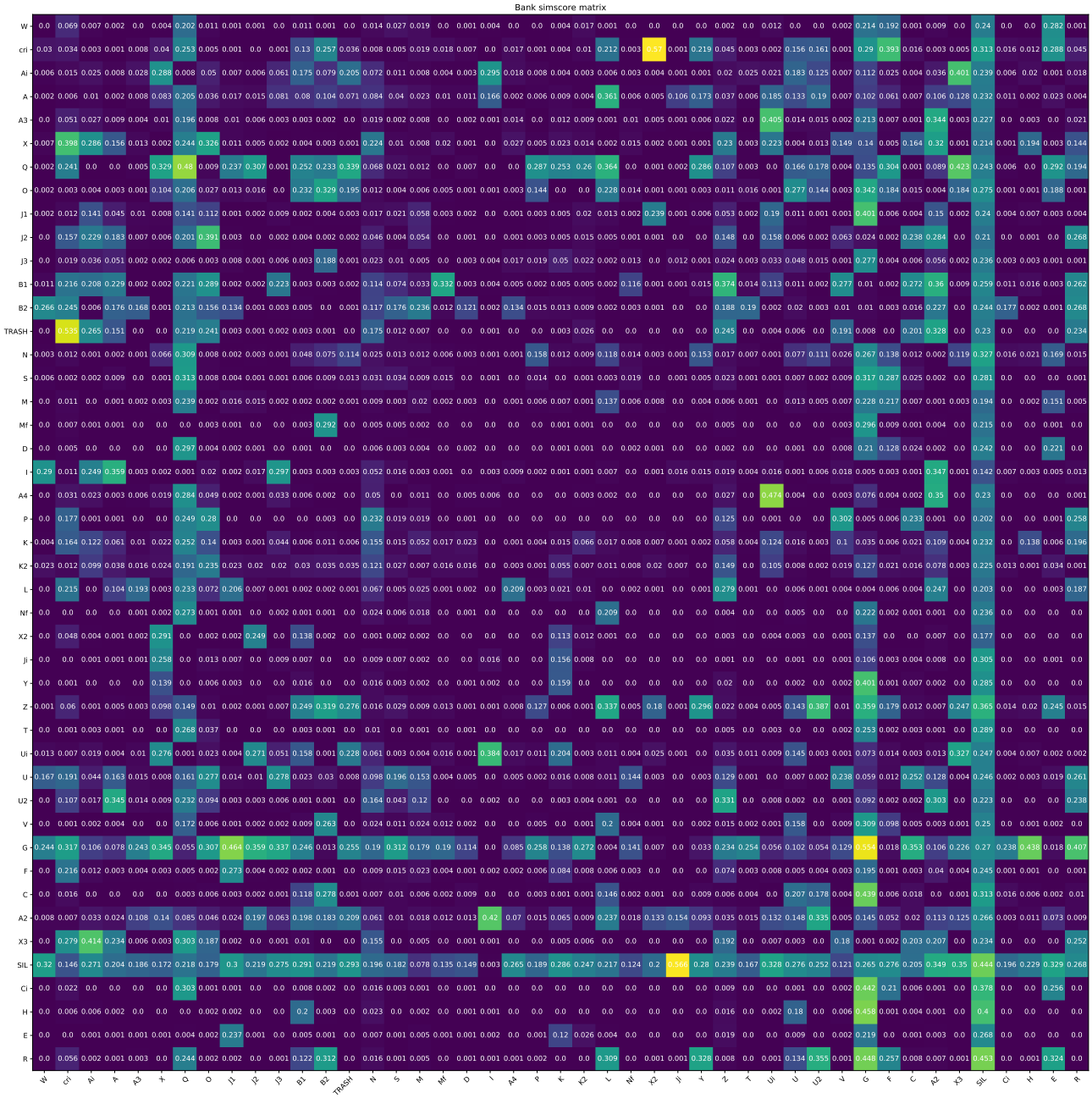



FIGURE 8 – Matrice de variance de la similarité pour la banque résumée