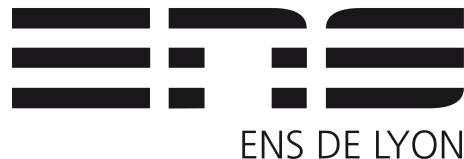


# CanaryMining : Analyse du chant des canaris et développement d'outils associés

**Vincent Gardies**

Encadré par Dr. Xavier Hinaut



Rapport de stage  
de fin de L3

au sein de l'équipe  
**Mmnemosyne**  
de l'institut de recherche  
**INRIA**  
du 19 juin au 28 juillet 2023

# Introduction

Ce rapport présente le stage de six semaines que j'ai effectué au sein de Mnemosyne, équipe de recherche de l'Institut National de Recherche en Informatique et en Automatique (INRIA) et spécialisée en Neurosciences Computationalles, à l'Institut des Maladies Neurodégénératives (IMN).

## Table des matières

<b>Introduction</b>	<b>1</b>
<b>1 Analyse du chant des canaris</b>	<b>2</b>
1.1 Introduction à l'étude du chant des oiseaux . . . . .	2
1.2 Particularités du canari . . . . .	3
1.3 Le chant du canari . . . . .	3
1.4 Annotation d'un chant de canari . . . . .	5
<b>2 Canapy</b>	<b>7</b>
2.1 Version déjà existante . . . . .	7
2.1.1 Pré-traitement des données . . . . .	7
2.1.2 Entraînement des modèles . . . . .	7
2.1.3 Premières utilisations . . . . .	9
2.2 Canapy-reborn . . . . .	10
2.2.1 Une nouvelle architecture . . . . .	11
2.2.2 Mes contributions en programmation . . . . .	12
2.2.3 Autres améliorations notables . . . . .	12
2.2.4 Préparation à la publication . . . . .	13
2.2.5 Utilisation de la nouvelle version . . . . .	14
<b>Conclusion</b>	<b>17</b>
<b>Bibliographie</b>	<b>17</b>
<b>Annexes</b>	<b>18</b>
2.3 Présentation de l'équipe Mnemosyne . . . . .	18

# 1 Analyse du chant des canaris

## 1.1 Introduction à l'étude du chant des oiseaux

Le temps est un facteur clef dans le travail d'interprétation et d'analyse constant que réalise n'importe quel cerveau. Les multiples informations reçues en permanence par celui-ci nécessitent un traitement permettant de rendre compte aussi bien de leurs caractéristiques spatiales (couleur, forme, fréquence sonore, pression et tension) que temporelles, à savoir dans quel ordre précis ces informations sont perçues.

L'exemple le plus flagrant d'une telle organisation temporelle se trouve dans les vocalisations animales, dont la variante la plus familière n'est autre que le langage humain.

Les vocalisations ont des rôles multiples au sein du règne animal : appeler, avertir, se reproduire, partager. Le fait qu'elles permettent des tâches aussi subtiles que la reconnaissance d'individus au sein d'une population renseigne sur la grande précision des mécanismes responsables du traitement des vocalisations. Le récepteur doit être capable de discriminer finement les sons de son espèce, tandis que l'émetteur doit posséder un contrôle parfait de ses organes vocaux. Il s'agit dans ce cas de maîtriser avec finesse les caractéristiques dites spatiales du son, à savoir les fréquences, puissances et harmonies qui le composent.

Mais ces caractéristiques spatiales seules ne peuvent suffire à décrire les vocalisations. Le fait qu'un mot écouté à l'envers ne soit reconnaissable, même lorsqu'il est composé d'une syllabe le prouve. Ces mots renversés ont pourtant les mêmes caractéristiques fréquentielles, la même puissance. Les sons doivent donc être arrangés dans le temps dans un ordre établi en syllabes, mots et phrases, pour ce qui est du langage humain, et dans toute une gamme d'autres motifs temporels pour ce qui est des autres espèces.

La langue française n'est ainsi constituée que d'un répertoire de 37 phonèmes, des sons identifiables uniquement par leurs caractéristiques spatiales. Mais en les combinant pour former des séquences temporelles, le français peut s'étendre sur un vocabulaire de plus de 35000 mots courants. Ces mots sont la conjugaison de phonèmes dans un ordre précis. De ce vocabulaire peut émerger une quantité infinie de séquences de mots, que l'on appelle phrases. La langue française se définit alors comme l'ensemble des vocalisations comprises et produites par les francophones et des combinaisons possibles de ces vocalisations, plus couramment appelées le lexique et la syntaxe.

Pour rendre compte de la structure temporelle des vocalisations d'un oiseau, il est alors nécessaire de faire apparaître :

- Un lexique ou répertoire, contenant l'ensemble des sons de bases de l'espèce (ou a minima de l'individu étudié).
- Une syntaxe, plus ou moins évoluée, c'est à dire des liens entre les unités ou groupes d'unités du répertoire.

## 1.2 Particularités du canari

Parmi les plus de 4500 espèces d'oiseaux chanteurs, il existe aussi bien des chants simples que complexes. Cependant, des informations quantitatives sur la complexité statistique du chant ne sont disponibles que pour quelques espèces. Le chant des oiseaux a souvent été décrit en termes de statistiques de transition de premier ordre, par exemple entre les syllabes adjacentes chez le diamant mandarin. Cependant, l'analyse du chant du diamant de Gould révèle également des dépendances non adjacentes où les probabilités de transition entre les syllabes dépendent non seulement de la syllabe actuelle, mais aussi d'une ou de plusieurs syllabes précédemment chantées. Formellement, cela implique que la syntaxe du chant doit être modélisée par une chaîne de Markov d'ordre deux ou supérieur.

En plus des études quantitatives détaillées sur la syntaxe chez les diamants de Gould en laboratoire, de nombreuses études sur le terrain ont décrit un éventail d'influences sur la prestation du chant. Par exemple, un stimulus auditif peut participer à la sélection d'éléments d'un répertoire vocal, que l'on observe chez le bruant des marais, qui trie son répertoire pour correspondre le plus possible au dernier chant produit par l'un de ses congénères.

Dans l'ensemble, de nombreux éléments suggèrent que les oiseaux chanteurs peuvent conserver une trace en mémoire des chants entendus ou chantés pendant au moins quelques secondes. Ainsi il existerait des corrélations à longue distance entre les points de décision dans le chant. Jusqu'où s'étend la mémoire des choix passés dans les chants les plus complexes ?

L'un des chanteurs les plus complexes qui peuvent être facilement élevés en laboratoire est le *canari domestique*. Leurs chants sont gouvernés par des corrélations à longue distance dans une syntaxe à plusieurs niveaux hiérarchiques. Le temps qu'un oiseau passe à répéter une syllabe ou la décision de ce qu'il chantera ensuite dépend de l'historique du chant, et les corrélations entre le passé et le présent peuvent s'étendre sur des durées allant jusqu'à 10 secondes, englobant 4 phrases ou plus constituées de dizaines de répétitions de syllabes.

Les expérimentations visant à étudier la production de séquences temporelles ordonnées de sons consistent alors généralement à enregistrer en parallèle les chants émis par l'oiseau et les signaux électrophysiologiques émis par certaines de ses aires cérébrales identifiées comme responsables de ses capacités vocales et auditives.

## 1.3 Le chant du canari

L'analyse de l'organisation séquentielle des vocalisations animales suppose donc de pouvoir identifier l'ensemble des sons ou groupes de sons formant le "lexique" de l'individu ou de l'espèce. Chez l'humain, cette capacité d'identification émerge durant l'enfance. Le cerveau est alors capable d'associer à chaque portion du flux audio l'entrée du lexique correspondante, sans effort. Une partie de l'intérêt de l'étude de l'organisation séquentielle des vocalisations animales



FIGURE 2 – Photo d'un canari domestique

réside d'ailleurs dans la recherche des mécanismes neurologiques qui permettent un tel apprentissage de la segmentation des sons.

Cette segmentation des sons en unités hiérarchisées – phonème, syllabe, mot, phrase – peut également être trouvée, sous d'autres formes, dans les vocalisations des oiseaux chanteurs, et en particulier du canari. Le canari présente trois niveaux de hiérarchie dans ses vocalisations :

- La syllabe, plus petite unité identifiable par l'humain, et supposément par le canari lui-même. Elle est dotée d'une signature fréquentielle unique la différenciant de toutes les autres syllabes du répertoire de l'oiseau. Chaque canari dispose d'un répertoire d'une quinzaine de syllabes, variable au cours de la vie de l'individu et de la période de l'année. Le canari apprend certaines syllabes de ses congénères. Ainsi des populations de canaris peuvent posséder un répertoire commun.
- La *phrase*, composée d'un unique type de syllabe, répété ou non. Une phrase peut ainsi contenir une unique copie d'une syllabe, ou plusieurs dizaines de copies contiguës dans le temps.
- Le *chant*, composé d'un ensemble de phrases. Les chants peuvent durer plusieurs dizaines de secondes et contiennent généralement de 10 à 20 phrases.

Compte tenu de la structure particulière du chant des canaris, composés de répétitions de motifs très distincts, il semble raisonnable de poser des heuristiques. Il est très probable que ces motifs – syllabes et phrases du canari – soient les unités de bases lui servant à construire son chant. Cette hypothèse est validée par les résultats expérimentaux de Markowitz et al (2013) où ces unités apparaissent clairement comme agencées suivant une syntaxe précise.

## 1.4 Annotation d'un chant de canari

Pour travailler sur les chants de canaris il faut dans un premier temps pouvoir identifier chaque composantes de celui-ci. Pour cela, il faut établir pour chaque spécimen un répertoire des différentes syllabes qu'il peut utiliser. Ce répertoire est arbitraire étant donné que, bien qu'il existe des similitudes entre certaines syllabes de différents canaris, les répertoires dépendent de l'entourage et de l'âge de ceux-ci. Il n'existe pas non plus de consensus concernant les noms attribués à chaque syllabe. Il est estimé que le répertoire d'un canari est composé de 15 à 25 syllabes distinctes. L'expérimentateur peut donc définir le répertoire d'un canari grâce à l'analyse audio mais aussi visuelle, en utilisant les spectres de fréquence de plusieurs chants de ce canari.

Cependant définir un répertoire est complexe. En effet certaines syllabes peuvent s'avérer très proches tout en ayant quelques différences. L'expérimentateur doit alors prendre un choix entre grouper les syllabes, créer deux classes distinctes, ne pas traiter ces syllabes. La première et la dernière option peuvent sembler néfastes pour la qualité de l'annotation. Ainsi l'expérimentateur a généralement tendance à créer davantages de classes, presque 40, alors qu'on peut en trouver entre 15 et 25 dans la littérature.

Une autre raison de cette complexité est le déséquilibre de représentation des syllabes dans la globalité des chants d'un même oiseau. La figure 3 montre que l'apparition des différentes syllabes du canari appelé Marron1 varie très fortement entre les différentes classes de syllabes. Dès lors pour définir un répertoire il est nécessaire de parcourir une très grande partie de l'ensemble des chants à disposition.

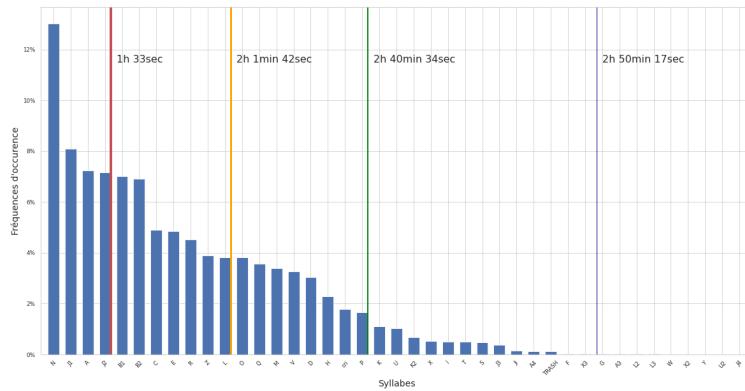


FIGURE 3 – Occurrences des différentes catégories de syllabes dans un des corpus annoté à la main (marron1)

Ensuite, un fastidieux travail de classification est nécessaire. Il faut déter-

miner, pour toutes les phrases du chant, le type de la syllabe qui est répétée. Ensuite il faut délimiter la durée précise de la phrase et annoter avec le nom que l'on a attribué à la syllabe. Un logiciel efficace pour effectuer ce travail est audacity (figure 4). Les chants les plus longs durent près de 45 secondes et les phrases varient entre 0.3 et 1.5 secondes.

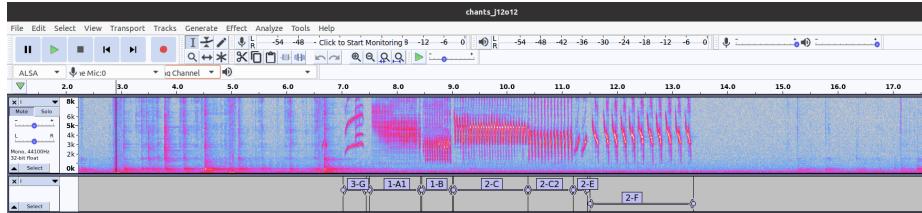


FIGURE 4 – Exemple d'annotation d'un chant de canari sur le logiciel audacity en utilisant les spectres audios

## 2 Canapy

Pour accélérer le processus d'annotation, le logiciel Canapy a été créé en 2020 par *Nathan Trouwain*, qui était à l'époque stagiaire chez Mnemosyne et y fait actuellement une thèse. Canapy est un annotateur semi-automatique utilisant de l'intelligence artificielle pour annoter les chants. Il est en accès libre sur GitHub (<https://github.com/birds-canopy/canapy>).

### 2.1 Version déjà existante

Les sections suivantes détaillent rapidement l'usage et le fonctionnement de la version de Canapy lors du début de mon stage.

#### 2.1.1 Pré-traitement des données

Dans un premier temps Canapy récupère des fichiers audios contenant chacun un unique chant de canari en format wav d'une part, et des fichiers csv contenant les annotations sous la forme d'un tableau contenant le temps du début et de fin d'une phrase ainsi que la syllabe répétée. Tout fichier audio doit être lié à un fichier csv contenant les annotations du chant en question et réciproquement. Ces fichiers sont présents dans un dossier qu'il faut préciser au lancement de Canapy.

Pour traiter efficacement les fichiers audios, Canapy effectue sur chacun d'eux une transformation selon le *Mel Frequency Cepstral Coefficients* (MFCC), un standard dans la représentation des vocalisations. Cette transformation permet de passer d'une piste audio échantillonée à 44100 Hertz, soit un long vecteur numérique où chaque seconde est codée par 44100 nombres décimaux, à trois vecteurs de 13 coefficients qui contiennent l'information compressée de la fréquence, sa dérivée première, et enfin sa dérivée seconde, pour un pas de temps d'environ 10 milisecondes. Cette manipulation divise par plus de 10 la quantité de données en perdant une précision négligeable. Cette transformation est réalisée grâce à la bibliothèque python Librosa. L'ensemble des mfcc est alors stocké dans un DataFrame (tableau à deux dimensions avec des colonnes nommées et particulièrement pratiques à utiliser) de la bibliothèque python Pandas. Dans ce grand DataFrame où chaque ligne contient les données mfcc d'un pas de temps, ainsi que le chant auquel elles correspondent, on ajoute l'annotation correspondante contenue dans le fichiers csv associé au chant en question. Cette annotation peut être l'une des annotations du répertoire, ou bien SIL qui correspond à un silence. Le DataFrame contient alors la totalité des données fournies à Canapy.

#### 2.1.2 Entraînement des modèles

Canapy utilise de l'intelligence artificielle et plus particulièrement des *Echo State Network* (ESN). Les ESN sont un type de réseaux de neurones artificiels

permettant l'apprentissage de tâches basées sur des séries temporelles, autrement dit de données organisées dans le temps. Il s'agit d'un type de réseau dit récurrent : le traitement des données appliqué prend en compte à la fois la donnée courante mais également les données précédemment vues. Dans le cas d'un chant de canari par exemple, un instant du chant sera annoté automatiquement par l'algorithme en prenant en compte les instants précédemment reçus par celui-ci, permettant d'intégrer la dimension temporelle au traitement. De plus, les ESN sont relativement légers en terme de temps et de puissance de calcul requis, en comparaison avec d'autres méthodes d'apprentissage automatique. Ils requièrent notamment une paramétrisation moins fine. Ils obtiennent néanmoins des résultats proches de ceux obtenus avec des méthodes plus complexes sur de nombreuses tâches. Les ESN peuvent être facilement utilisés grâce à la bibliothèque python ReservoirPy, développée par Xavier Hinaut, puis par Nathan Trouvain.

Dans le but de maximiser les chances de résultat, deux modèles sont entraînés sur les données citées ci-avant. Ces modèles doivent tout deux prédire quelle syllabe est associée à chaque pas de temps de 10 milisecondes défini par les MFCC. La puissance des ESN réside alors dans leur capacité à retenir l'information des pas de temps précédemment prédits pour assister la prédiction du pas de temps suivant. Compte-tenu de cette propriété de récurrence, deux modèles utilisant deux tâches d'apprentissage différent sont utilisés :

- Modèle *syntaxique* : cet ESN est entraîné en utilisant des chants complets. La propriété de récurrence est exploitée sur toute la durée du chant. L'ESN est donc confronté à des données "réelles", et peut retenir des informations sur toute la durée du chant. La "mémoire" de l'ESN est réinitialisée entre chaque chant analysé.
- Modèle *non-syntaxique* : cet ESN n'est pas entraîné en utilisant des chants complets, mais uniquement des phrases (groupes de syllabes répétées), mélangées sans suivre l'ordre du chant. La propriété de récurrence n'est exploitée que sur la durée d'une phrase. La "mémoire" du réseau est réinitialisée entre chaque phrase. L'ESN doit donc exploiter uniquement les caractéristiques d'une phrase pour la reconnaître, sans possibilité d'accéder aux phrases précédemment perçues.

Ainsi ces deux modèles ont des points forts, l'un permet de mieux intégrer des dépendances linéaires dans le chant du canari (modèle syntaxique) tandis que l'autre palie le problème du déséquilibre de la représentation des syllabes dans l'ensemble des chants de canaris (voir figure 3).

Pour compléter ces deux modèles, Canapy en utilise un dernier permettant de les combiner : le modèle *Ensemble* (figure 5). Ce modèle utilise le *hard-vote* sur les prédictions des modèles syntaxique et non-syntaxique pour construire une troisième série de prédictions. Le hard-vote peut être vu comme un système de vote à l'applaudimètre : pour chaque pas de temps du chant annoté par les modèles syntaxique et non-syntaxique, le modèle ensemble retient la prédiction la plus forte entre les deux modèles. La "force" de la prédiction est quantifiable, car ces prédictions sont des vecteurs de nombres réels. Entre les deux

vecteurs prédits, on sélectionne donc la valeur maximale des valeurs maximales de chaque vecteur. La position de cette valeur dans le vecteur représente alors la syllabe prédictée. Le modèle ensemble permet donc de sélectionner uniquement les prédictions pour lesquelles les modèles sont les plus "sûrs d'eux".

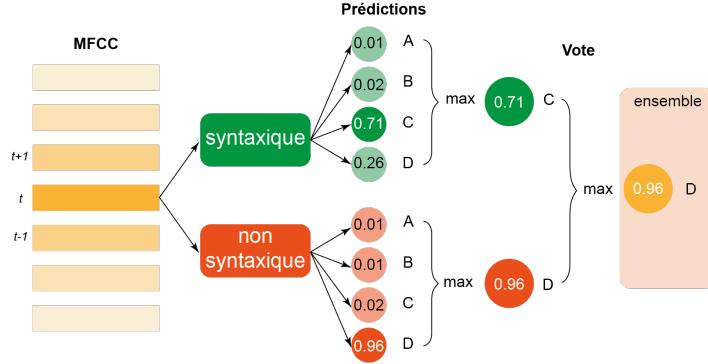


FIGURE 5 – Hard vote pour la construction du modèle ensemble

### 2.1.3 Premières utilisations

Une des premières tâches qui m'a été confiée lors de mon stage a été d'entraîner les modèles syntaxiques, non-syntaxiques et ensemble sur les données du canari Rouge6. Ces données avaient préalablement été annotées par *Alizée Fournier*, stagiaire de l'équipe Mnemosyne.

La première étape a donc été d'installer Canapy. Un manuel d'utilisation sous Microsoft avait déjà été réalisé par *Alizée*. Il m'a donc été proposé d'en créer un pour les utilisateurs de Linux, en m'a aidant de celui déjà réalisé. Cependant j'ai eu quelques difficultés, en effet Canapy a été conçu il y a plusieurs années et n'a pas reçu de maintenance pour assurer son fonctionnement sur les nouvelles versions de Python et de ses librairies. Face à trop de problèmes sur un code conséquent avec lequel je n'étais pas familier j'ai opté pour une rétrogadation des versions de mes modules, et de Python (en 3.6) pour les ajuster sur celles de la création de Canapy. J'ai alors pu récupérer les données annotées et lancer le logiciel (Figure 6).

Je me suis ensuite entraîné à corriger les annotations, fusionner les classes qui me semblaient les plus pertinentes directement sur Canapy. Pour cela je m'aaidais de la matrice de confusion des différents modèles ainsi que des échantillons de chaque phrase que je pouvais écouter ou même visualiser grâce au spectre (figure 7). Ces manipulations sont utiles pour repérer les erreurs dans l'annotation faite à la main comme une erreur de typographie lors de l'attribution de la classe de syllabe d'une phrase, ou bien d'une erreur dans l'attribution de la classe. Il est aussi fréquent de fusionner deux classes qui sont trop proches, ou pas suffisamment distinctes pour être reconnues à chaque fois par Canapy. Enfin j'ai récupéré les

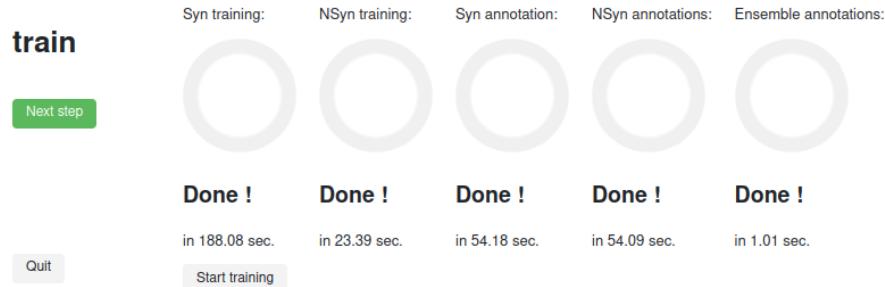


FIGURE 6 – Interface de Canapy après l’entraînement des modèles

différentes métriques de performance des modèles tels que le f1-score, le macro-avg, le weighted-avg pour discuter de leurs pertinences avec *Xavier Hinaut* et *Nathan Trouvain*.

J’ai ensuite réitéré l’opération en séparant au préalable une partie des données annotées, correspondant à 10 chants. J’ai alors entraîné des modèles sur Canapy avec le reste des chants. J’ai récupéré les modèles créés, et ai écrit un script python pour créer une annotation des 10 chants tests grâce aux modèles, afin de la comparer avec l’annotation faite à la main. Pour les comparer j’ai utilisé une métrique très simple, l’accuracy, c’est-à-dire que le script renvoie le pourcentage de pas de temps qui ont été annotés avec la même syllabe. J’ai obtenu des scores surprenamment hauts, 96 % de correspondance. Après discussion avec mes encadrants, nous avons abouti au fait que j’avais fusionné trop de classes de syllabes pendant la correction des annotations, et alors perdu beaucoup de précision dans les annotations, ce qui explique un score aussi haut. Effectivement les scores précédemment obtenus étaient plutôt entre 80 et 85 %.

## 2.2 Canapy-reborn

Une nouvelle publication de Canapy étant prévue il a été décidé au cours de mon stage de rénover Canapy. Cette décision a été motivée par plusieurs raisons :

- La *lisibilité* du code, en effet le code était peu séparé en terme de fichiers, et était donc très dense et dur à lire par une personne extérieure
- L'*efficacité* du code, parce que certaines parties du traitement des données ne sont pas gérées au mieux
- La *compatibilité* avec les versions actuelles et futures, en effet Canapy ne pouvait pas être utilisé avec Python $\geq 3.8$ , et de même avec de nombreuses librairies
- La *maintenabilité* du code, si tout ces points étaient améliorés, Canapy deviendrait maintenable par d’autres équipes

C’est pourquoi le projet Canapy-Reborn, auquel j’ai participé avec *Nathan*

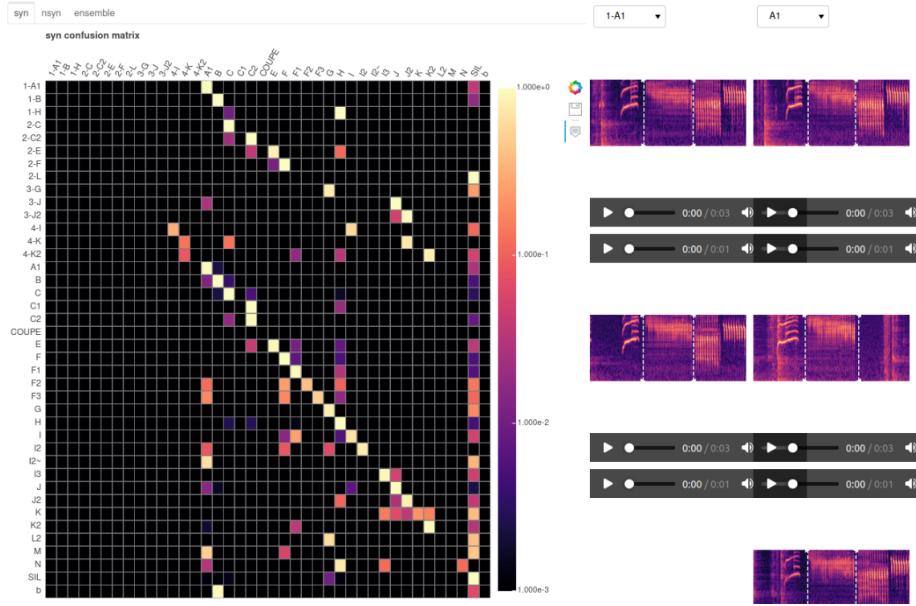


FIGURE 7 – Interface de Canopy pendant la correction des annotations. A droite la matrice de confusion du modèle syntaxique, à gauche les échantillons de deux classes

*Trouvain*, est né. Ce logiciel est disponible sur GitHub (<https://github.com/birds-canopy/canopy-reborn>).

### 2.2.1 Une nouvelle architecture

La strucure principale choisie est View-Controller-Model. Celle-ci correspond au fait de séparer l'ensemble des fonctions et classes qui transforment et manipulent les données dans une partie que l'on nomme le Model, et qui doit être relativement facile à utiliser directement avec un script python ou dans un terminal. La View correspond à toute la partie interface graphique du logiciel, et donc les affichages des résultats, les boutons, la progression des calculs, etc. Le Controller est chargé de faire le lien entre ces deux parties, qui ne doivent pas être directement "en contact".

L'une des principales améliorations qui a été décidée porte sur l'utilisation de classes mieux définies et mieux organisées. Tout d'abord les jeux de données sont sauvegardés dans une classe appelée Corpus. Ce corpus contient le DataFrame avec les données principales, mais aussi la configuration utilisée ainsi que les transformations que le corpus a pu subir, entre autres.

Ensuite une nouvelle classe, Transform, est héritée par deux sous-classes SynTransform et NSynTransfrom. Les instances de celles-ci prennent en argument

un corpus, et retournent un corpus avec l'ensemble des modifications nécessaires pour que ce corpus soit traité par un Annotator.

Les Annotator sont une nouvelle classe qui contiennent principalement les modèles des ESN, mais aussi un booléen permettant de savoir s'ils sont déjà entraînés. Trois sous-classes héritent de cette classe : les SynAnnotator, les NSynAnnotator, et Ensemble. Chacune contient les fonctions adéquates pour respectivement s'entraîner et prédire selon un modèle syntaxique, non-syntaxique ou ensemble.

Enfin les parties View et Controller, ont essentiellement été retravaillées par *Nathan Trouvain*.

### 2.2.2 Mes contributions en programmation

*Nathan Trouvain*, qui s'y connaît davantage aussi bien en python qu'à propos de Canapy, a été le principal développeur de Canapy Reborn. Cependant, nous avons organisé à plusieurs reprises des séances de *pair-programming*, pendant lesquelles l'un de nous deux codait et l'autre l'orientait, posait des questions, ou proposait de nouvelles idées pour se mettre d'accord à propos de la forme du code. Ces sessions se sont révélées cruciales pour que je comprenne la rigueur et la syntaxe attendues dans le code.

Une fois familiarisé avec ces différents éléments des portions de codes, des tâches de plus en plus importantes m'ont été confiées. J'ai commencé en autonomie par l'entraînement de l'annotateur non-syntaxique, qui était relativement proche de celui de l'annotateur syntaxique qui avait été réalisé ensemble plus tôt. Puis j'ai pu m'occuper de certaines transformations qu'il fallait appliquer au corpus. Ne connaissant pas la bibliothèque Pandas ou Librosa, je devais en même temps me renseigner sur celles-ci pour maîtriser les outils utilisés dans Canapy. J'ai enfin dû travailler sur l'import et l'export de configurations. Une partie particulièrement complexe a été la rétrocompatibilité des modèles créés avec l'ancienne version de canapy. Il a fallu utiliser le module Pickler, puis créer notre propre classe d'unpickler pour parvenir à traiter les anciens annotateurs. J'ai aussi essayé de travailler sur la rétrocompatibilité des configurations mais des difficultés sont survenues, et par manque de temps nous avons décidé de passer sur des points plus urgents.

### 2.2.3 Autres améliorations notables

En plus des différentes améliorations évoquées de Canapy Reborn par rapport à sa version précédente, il est judicieux de mentionner l'utilisation du module python Poetry. Ce module permet de gérer efficacement l'ensemble des dépendances entre les modules utilisés dans canapy. Il permet notamment de créer un environnement dans lequel les versions des modules permettent de faire tourner le code sans problème. Le dernier point sur lequel poetry est un véritable progrès est l'installation de canapy. En effet, poetry met en place une installation très guidée et simple de tous les modules nécessaires. Ainsi les problèmes

que j'avais eu en essayant d'installer canapy sur mon ordinateur ne devraient plus exister.

Une autre amélioration importante est l'utilisation du module Crowsetta. Ce module python a été développé par David Nicholson. Il s'agit d'un outil pour travailler avec un large éventail de formats pour les vocalisations animales. L'intégrer dans cette nouvelle version de Canapy est un moyen à la fois de standardiser le logiciel, mais aussi de mettre en avant Crowsetta, dans l'espoir d'arriver à un consensus dans la sphère de la recherche au niveau des formats, noms et méthodes. Ce dernier point est crucial pour le partage et la communication de données ou résultats entre les différentes équipes travaillant sur la vocalisations animales à travers le monde.

#### 2.2.4 Préparation à la publication

En vue de la publication du logiciel Canapy, il a été nécessaire de commenter rigoureusement le code ainsi que de créer de nombreux tests pour vérifier le bon fonctionnement des outils développés. Cette partie m'a été confiée une fois la partie Model quasiment terminée. Ayant travaillé sur une bonne partie des codes de cette partie, j'étais apte à commenter la plupart. J'ai dû apprendre le format rigoureux des docstrings de numpy pour les appliquer à la nouvelle version de Canapy. En parallèle je réalisais les tests basiques des fonctions développés et les intégrais dans la docstring ou dans des fichiers réservés aux tests.

```
1      """
2          Create a Corpus object from audios, annotations, and
3          spectrogram files stored on the disk.
4
5          Parameters
6          -----
7              audio_directory : str, optional
8                  Path of the directory that contains the audio tracks.
9              spec_directory : str, optional
10                 Path of the directory that contains the spectrogram
11                 files.
12              annots_directory : str, optional
13                  Path of the directory that contains hand-made
14                  annotations.
15              config_path : str, optional
16                  Path of the directory that contains the configuration.
17                  By default, default_config (from config.py or config.
18                  toml) will be applied.
19              annot_format : str, default="marron1csv"
20                  The format of the annotation data.
21              time_precision : float, default=0.001
22                  The time precision.
23              audio_ext : str, default=".wav"
24                  The extension for the audio files in the
25                  audio_directory.
26              spec_ext : str, default=".mfcc.npy"
27                  The extension for the spectrogram files in the
28                  spec_directory.
```

```

23
24     Returns
25
26     Corpus
27         Corpus object that contains the audio, annotations, and
28         spectrogram files from the given directories.
29
30     Raises
31         ValueError
32             At least one of audio_directory or spec_directory must
33             be provided.
34
35     Notes
36
37     Examples
38
39         >>> corpus_audio = Corpus(audio_directory="/home/
40             vincent/Documents/data_canary/audio")
41             # corpus_audio is a corpus with only audio tracks
42
43         >>> corpus_annotation = Corpus(
44             >>>     audio_directory="/home/vincent/Documents/
45                 data_canary/mix_audio_annot",
46             >>>     annots_directory="/home/vincent/Documents/
47                 data_canary/mix_audio_annot"
48             >>> ) # This corpus is made with the audio and
49             annotation files in the 'mix_audio_annot' folder
50
51         """
52
53

```

J'ai aussi été chargé de rédiger le manuel d'utilisation de canapy sous la forme d'un JupyterNotebook, qui sera publié sur GitHub avec la nouvelle version. J'ai pu réaliser ces tâches relativement longues avec peu de supervision.

### 2.2.5 Utilisation de la nouvelle version

Lorsque la version était quasiment terminée, il a fallu la tester avec de grosses quantités de données, en conditions réelles. J'ai alors été chargé de rassembler les données sur un disque dur de l'équipe Mnemosyne, où les annotations et les fichiers wav étaient quelques peu dispersés, et sous un format qui n'était pas utilisable par canapy (aup3, le format d'Audacity $\geq 3.0$ ). J'ai alors rédigé un script quasiment automatique pour rassembler les fichiers et les reformater correctement.

J'ai lancer Canapy et j'ai pu entraîner différents modèles grâce à des datasets de 30 à 80 chants pour 4 oiseaux : Vert4, Rouge18, Argent11, Argent19. Pour chaque oiseau j'ai entraîné les 3 modèles grâce à Canapy-Reborn, puis j'ai effectué diverses corrections au niveau des classes, pour optimiser la performance du modèle. J'ai ensuite récupéré les différentes métriques pour les stocker dans le disque dur de l'équipe.

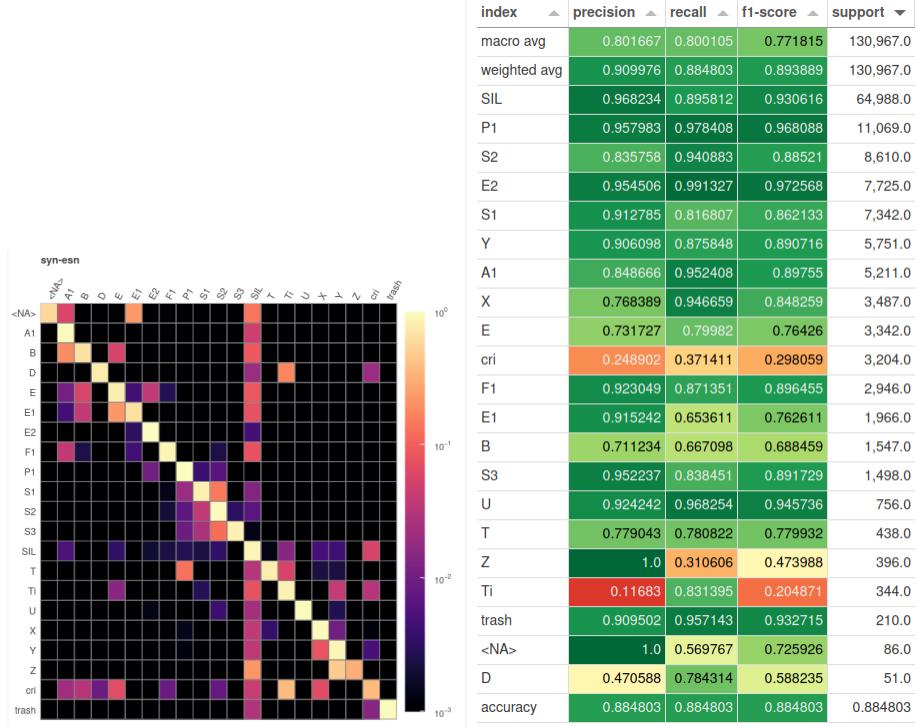


FIGURE 8 – Matrice de confusion et métriques par syllabes de l'annotateur syntaxique en mode train pour l'oiseau Argent19 avec 55 chants

J'ai ensuite voulu montrer l'amélioration des performances du modèles en fonction de l'augmentation de la taille du set de données. Pour cela j'ai regroupé 10 chants pris au hasard que j'ai désigné comme étant les chants "tests". Puis à l'aide de script python que j'ai réalisé j'ai calculé les sommes partielles de la durée des chants complets restant les uns à la suite des autres. En choisisissant ensuite le séquençage, c'est-à-dire l'intervalle correspondant à la prise de deux mesures, on peut alors lancer le script, qui pour chaque mesure, va prendre le plus de chants complets sans dépasser le nombre imposé de secondes, puis va avec cette quantité de donnée, entraîner les trois modèles classiques d'ESN, pour comparer sur chaque chant "test" la *segment error rate* entre la prédiction des annotateurs et celle faite à la main (figure 9).

Enfin j'ai réalisé à partir du même script que précédemment, une autre étude concernant les métriques sur chaque classe. En se rappelant bien que l'annotateur a été bon si la métrique est proche de 1 et médiocre si elle est proche de 0. Cette fois ci, chaque ligne représente l'évolution de la mesure f1 d'une classe de syllabes. La précision et les moyennes ont aussi été rajoutées (figure 10).

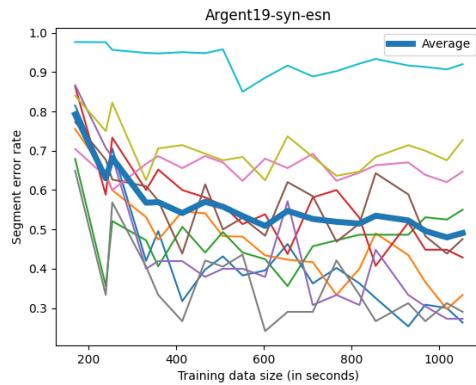


FIGURE 9 – Variation du Segment Error Rate en fonction de la durée du dataset d’entraînement sur un annotateur syntaxique et avec les chants de Argent19 avec 10 chants tests

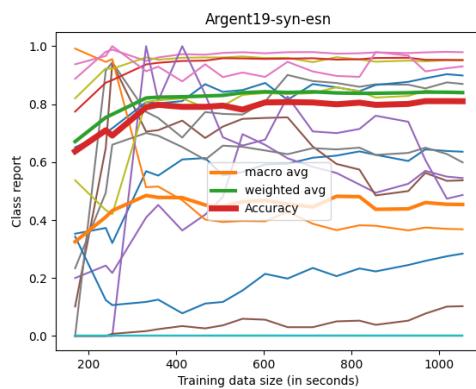


FIGURE 10 – Variation du f1-Score en fonction de la durée du dataset d’entraînement sur un annotateur syntaxique pour toutes les classes de syllabe des chants de Argent19 avec 10 chants tests

## Conclusion

Pour conclure, ce stage s'est globalement bien passé. Je remercie l'équipe Mnemosyne pour son accueil mais toutes celles que j'ai pu cotôyer. Je remercie tout particulièrement Xavier Hinaut et Nathan Trouvain pour l'aide, et les conseils fournis tout au long de mon stage. Les objectifs principaux fixés au début du stage, tels que la rénovation de Canapy, ont dans l'ensemble été atteints. Le temps m'a cependant manqué vers la fin du stage pour continuer à analyser les performances de canapy avec plus de précision, ou bien pour finaliser quelques fonctionnalités sur Canapy-Reborn.

## Bibliographie

Long-range Order in Canary Song, Jeffrey E. Markowitz, Elizabeth Ivie, Laura Kligler, Timothy J. Gardner, May 2, 2013 <https://doi.org/10.1371/journal.pcbi.1003052>

Analyse de la structure du chant de canaris domestiques et développement des outils associés, Nathan Trouvain, 2020

Etude de la structure du chant des Canaris pour l'étude du langage humain, l'analyse et la création d'un protocole du logiciel Canapy, Alizée Fournier, 2023

Cazala, A. (2019). Codage neuronal de l'ordre des signaux acoustiques dans les chants des oiseaux (Doctoral dissertation, Université Paris Saclay (COmUE)).

## Annexes

### 2.3 Présentation de l'équipe Mnemosyne

Les Neurosciences Computationnelles forment un vaste champ de recherche à la croisée de l'informatique, des mathématiques et des neurosciences. Elles visent à concevoir des algorithmes permettant de simuler et modéliser le fonctionnement du système nerveux. Mnemosyne étudie ainsi diverses fonctions cognitives du cerveau humain et animal, allant du traitement des sens et de l'activation des muscles jusqu'à la modélisation de comportements sociologiques, autour de grandes thématiques comme le langage ou l'éducation. Ces simulations et modèles complètent et sont complétés par les données issues de l'expérimentation en neurosciences, et offrent de nouvelles perspectives d'analyse, d'interprétation et de recherche au travers du prisme des sciences de l'information, de la modélisation mathématique et de l'intelligence artificielle.

Les réunions hebdomadaires avec toute l'équipe, et la présence d'un Open-Lab présentant les recherches et sujets d'intérêts de toute l'équipe ont été des supports très utiles pour me renseigner sur le domaine de compétence de chacun.