Rapport de stage

Apprentissage de structures multi-dimensionnelles pour l'improvisation musicale

Ken Déguernel

Inria Nancy Grand Est

Stage encadré par : Emmanuel Vincent et Gérard Assayag

ken.deguernel@inria.fr

7 août 2015

Résumé / Summary

L'improvisation automatique est un domaine récent issu des études sur la créativité artificielle. Les systèmes actuels d'improvisation automatique sont capables de prendre en considération des informations unidimensionnelles, typiquement la mélodie, fournies en direct par un musicien ou lues dans un corpus afin de générer de nouvelles improvisations par une recombinaison du matériau musical. Cependant, ils ne sont pas capables de prendre en considération des informations multi-dimensionnelles. Or la musique comporte de nombreuses dimensions : mélodie, harmonie, rythme... Dans ce rapport, nous évaluons l'intérêt de prendre en compte plusieurs dimensions musicales dans le cadre de l'improvisation en appliquant des méthodes d'interpolation de sous-modèles issues du domaine de l'apprentissage automatique et des principes de modélisation de langage. Nous effectuons l'apprentissage et les tests sur un corpus d'improvisation de jazz. Nous évaluons les résultats en comparant leurs entropies croisées. Nous montrons que dans certains cas, l'interpolation de sous-modèles permet d'obtenir un meilleur pouvoir de prédiction que lorsque l'on ne considère qu'une seule dimension musicale.

Mot-clefs : improvisation automatique, apprentissage automatique, modélisation du langage.

Automatic improvisation is a new field based on the study on artificial creativity. Current systems for automatic improvisation are able to take single dimensional information into account, usually the melody, played by a musician or read from a corpus to create new improvisations by a recombination of the musical content. However, they are not able to take multi-dimensional information into account. Yet music is multi-dimensional : melody, harmony, rhythm... In this thesis, we evaluate whether considering multi-dimensional information can improve current automatic improvisation systems. To do so, we use sub-model interpolation methods from the field of machine learning and language modelling. Training and tests are carried out on a jazz improvisation corpus. The models are compared in terms of their cross-entropy. We show that in some cases, sub-model interpolation provides better prediction power than single dimensional models.

Keywords : automatic improvisation, machine learning, language models.

Remerciements

Tout d'abord, je tiens à remercier Emmanuel Vincent et Gérard Assayag pour m'avoir encadré lors de ce stage. Merci pour les nombreux conseils. Et merci d'avance pour les nombreux conseils à venir ! Merci également à l'ensemble des membres du projet DYCI2.

Merci à mes collègues de bureaux à Inria Nancy : Juan Andrés Morales Cordovilla et Sunit Sivasankaran. Merci également à l'ensemble de l'équipe Multispeech.

Merci à Alain Dufaux, Jérome Nika et Rémi Fox pour cette merveilleuse rencontre avec l'EPFL au festival de Montreux.

Merci aux amis d'ATIAM avec qui j'ai passé une de mes plus belles années. Merci en particulier à Damien, Axel, Léna, Hugo.

Merci bien évidement à mes parents, à Leslie, Angie, Thomas, Guillaume, Louis et Pascal pour leur soutien permanent, l'intérêt qu'ils portent à mes projets, et sans qui l'univers serait bien moins amusant.

Et de manière plus générale, merci à toutes les personnes que j'ai côtoyées durant ce stage. Grâce à vous tous, je me serai bien amusé !

Libertas per cultum. That is all.

Table des matières

1 Introduction

L'improvisation automatique est un domaine récent issu des études sur la créativité artificielle. L'objectif principal est de mettre en place des interactions musicales improvisées entre des humains et des ordinateurs. De nombreuses problématiques sont mises en avant par ce domaine, comme par exemple la nécessité de réaliser un apprentissage interactif en temps réel, une écoute artificielle efficace, de modéliser des structures musicales, *etc.* Nous visons à créer une nouvelle pratique musicale par l'extension de la recherche sur l'improvisation débouchant sur la création d'interactions nouvelles entre humains et machines. Le projet DYCI2 s'interesse au cœur même de ces problématiques, et se base sur des études et des projets développés à l'Ircam par Gérard Assayag et les "OMax Brothers".

Ce stage a eu lieu dans le cadre du projet ANR DYCI2 (Dynamiques Créatives de l'Interaction Improvisée). Ce projet a pour objectif de créer des modèles efficaces d'écoute artificielle, d'apprentissage et de création automatique de musique permettant à des agents numériques d'avoir des interactions musicales improvisées en temps réel. Ce projet est séparé en trois tâches principales qui seront effectuées en parallèle :

— écoute informée créative,
— apprentissage interactif de structures musicales,
— dynamiques d'interaction improvisée.

Ce stage concerne la partie «Apprentissage interactif de structures musicales» de ce projet. L'objectif de cette tâche est de concevoir des méthodes d'apprentissage automatique sur des données symboliques permettant de représenter les corrélations entre différentes dimensions musicales (par exemple : hauteurs, harmonie, timbres...), et de reconnaître des structures multi-échelles émergeant dans un contexte d'improvisation.

Les systèmes actuels d'improvisation automatique sont capables de prendre en considération des informations unidimensionnelles, typiquement la mélodie, fournies en direct par un musicien ou lues dans un corpus afin de générer de nouvelles improvisations par une recombinaison du matériau musical. Ces improvisations peuvent être guidées par l'utilisation de paramètres de contrôle, par une écoute active de l'environnement musical en temps réel, ou par le suivi d'un scénario prédéfini. Cependant, aucune des méthodes actuelles ne prend en considération une combinaison des différentes dimensions musicales pour générer une improvisation. Prendre en compte cet aspect multi-dimensionnel de la musique semble très intéressant car cela correspond au comportement réel des improvisateurs.

L'objectif de ce stage est de s'intéresser aux méthodes d'apprentissage automatique basées sur l'interpolation de sous-modèles probabilistes pour l'aspect multi-dimensionnel de la musique afin d'évaluer l'intérêt de telles méthodes pour des tâches relatives à l'improvisation.
Ce stage s'est déroulé à Inria Nancy - Grand Est, dans l'équipe-projet Mul-

tispeech. Cette équipe est spécialisée dans la modélisation de la parole et du langage naturel et s'intéresse notamment à la modélisation statistique de la parole en se basant sur des modèles bayésiens et des réseaux de neurones profonds.

Dans ce rapport, après une brève présentation d'Inria et de l'équipe Multispeech dans la partie 2, nous effectuerons un état de l'art concernant les principaux logiciels d'improvisation automatique et les méthodes qu'ils emploient, puis sur les méthodes d'apprentissage par interpolation de sous-modèles dans la partie 3. Puis, nous appliquerons ces méthodes pour des tâches liées à l'improvisation automatique dans la partie 4. Nous finirons par une conclusion et une présentation des perspectives dans la partie 5.

2 Présentation de l'organisme d'accueil

2.1 Inria Nancy

Le centre de recherche Inria Nancy - Grand Est fut crée en 1986. Il développe l'essentiel de ses activités scientifiques en partenariat avec le CNRS, l'Universités de Lorraine, l'Université de Strasbourg et l'Université de Franche-Comté. Il accueille actuellement environ 220 chercheurs, doctorants et ingénieurs.

Le champ de recherche d'Inria Nancy est très large. Il s'intéresse à des thématiques informatiques variées avec 23 équipes-projets travaillant autour de cinq domaines principaux :

- mathématiques appliquées, calcul et simulation,
- algorithmique, programmation, logiciels et architectures,
- réseaux, systèmes et services, calcul distribué,
- perception, cognition, interaction,
- santé, biologie et planète numériques.

2.2 L'équipe Multispeech

Pour ce stage, je fut accueilli par l'équipe-projet Multispeech. Dirigée par Denis Jouvet, cette équipe-projet est constituée d'une trentaine de personnes (dont huit doctorants) d'Inria, de l'Université de Lorraine et du CNRS.
Ses recherches concernent le traitement de la parole, avec une attention particulière pour les aspects multisources (séparation de sources, reconnaissance robuste de la parole), multilingues (apprentissage de langues étrangères) et multimodaux (synthèse audiovisuelle). Ses recherches s'orientent selon trois axes :

- la modélisation explicite de la parole, en se basant sur des modèles physiques,
- la modélisation statistique de la parole, utilisant des techniques d'apprentissage automatique comme, par exemple, des modèles bayésiens et des réseaux de neurones profonds,
- la prise en compte des incertitudes liées à la forte variabilité du signal.

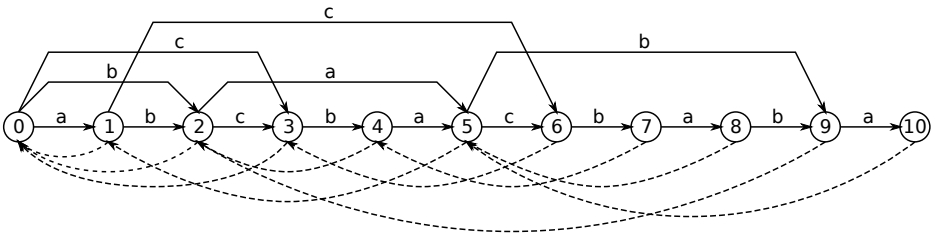

FIGURE 1 – Exemple d'oracle des facteurs pour le mot $w = abcbacbaba$. Les flèches pleines correspondent aux transitions de l'automate. Les flèches en pointillés sont des flèches de construction appellées liens suffixes.

3 État de l'art

3.1 Improvisation automatique

Les travaux sur l'improvisation automatique sont issus initialement de la recherche sur la composition automatique. De nombreux systèmes ont été developpés. Conklin [11], et Pachet et al. [24] développent des méthodes probabilistes basées sur des chaines de Markow cachées. D'un autre côté des méthodes basées sur la structure d'oracle des facteurs sont développés par les "OMax Brothers" [5, 32], Donze et al. [12], Schankler et al. [29]... Nous présentons ici trois logiciels issus de l'Ircam qui montrent bien les principes de l'improvisation automatique.

3.1.1 OMax

OMax est un logiciel d'improvisation automatique utilisant Max/MSP pour faire des prestations musicales en temps réel. Son principe est d'apprendre en direct le style d'un musicien et de générer une improvisation par une recombinaison du matériau musical fourni par le musicien. OMax ne fait appel à aucune notion préalable ni aucune règle harmonique issue d'une quelconque théorie musicale, ce qui lui permet de s'adapter à tous types d'improvisation. OMax se contente de modéliser le style de l'improvisateur en extrayant des règles implicites données par l'improvisation. Il repose sur le principe de *réinjection stylistique* [20], le musicien réagissant à son propre style au cours de l'improvisation. Son fonctionnement se base sur l'oracle des facteurs [5, 3] ; un automate issu de la bioinformatique utilisé pour la recherche de sous-chaînes. OMax peut prendre en entrée du MIDI polyphonique (*Musical Instrument Digital Interface* : protocole de communication pour instruments de musique électronique et pour ordinateurs), et de l'audio monophonique [4]. Dans le cas de l'audio, les hauteurs des notes sont détectées par l'algorithme Yin [10]. OMax possède une interface graphique permettant de visualiser l'oracle des facteurs en temps réel [18, 19].

La structure d'oracle des facteurs a été introduite par Crochemore et al. [2] pour effectuer la recherche de sous-chaînes, puis a été étendue au calcul de répétition et à la compression de données par Lecroq et Lefebvre [16]. Il s'agit d'une structure représentant au moins tous les facteurs d'un mot w. On dit que

x est un facteur de w si il existe deux mots y et z tels que $w = yxz$.

L'algorithme 1 décrit la construction de l'oracle des facteurs d'un mot $w = x_1...x_n$ [5, 16]. On note δ l'ensemble des transitions, et S l'ensemble des liens suffixes. On dit que x est un suffixe de w si il existe un mot y tel que $w = yx$. Les liens suffixes sont des flèches utilisées lors de la construction de l'oracle des facteurs, et lient chaque état avec l'état précédent partageant le suffixe commun le plus long. La notation $\delta(i, \sigma) \leftarrow j$ signifie qu'on crée une transition de i vers j étiquetée par σ. De même, $S(j) = i$ signifie qu'on crée un lien suffixe allant de j à i. On appelle liens facteurs l'ensemble des transitions crées pendant la construction de l'oracle allant d'un état i vers un état $i + k$ ($k > 1$).

```
Créer un état 0
S(0) ← −1
pour i allant de 1 à n faire
    Créer un état i
    δ(i − 1, x_i) ← i
    k ← S(i − 1)
    tant que k > −1 et δ(k, x_i) est indéfini faire
        δ(k, x_i) ← i
        k ← S(k)
    fin
    si k = −1 alors
        S(i) = 0
    sinon
        S(i) = δ(k, x_i)
    fin
fin
```

Algorithme 1 : Construction de l'oracle des facteurs.

De par sa construction, l'oracle des facteurs est un automate acyclique constitué de $n + 1$ états (n étant la longueur du mot w) et au plus $2n - 1$ transitions ($3n-2$ flèches en comptant les liens suffixes). La construction de l'oracle des facteurs se fait en $O(n)$ en temps et en mémoire. Un facteur est reconnu par l'oracle si il existe un parcours partant de l'état 0 et utilisant des transitions succesives de l'oracle épelant ce facteur. Les liens suffixes sont des flèches de constructions et ne peuvent pas être utilisées lors du parcours classique de l'oracle.
La Fig. 1 montre un exemple d'oracle des facteurs pour le mot $w = abcbacbaba$. L'ensemble des facteurs de w sont reconnus par un parcours dans l'oracle. Cependant, la structure d'oracle des facteurs n'est pas exacte dans le sens où il existe des facteurs reconnus par l'oracle, mais n'apparaissant pas dans w. Par exemple, le facteur *bacb* est reconnu par le parcours de la tableau 1, cependant, on peut remarquer que le facteur *aba* est reconnu par l'automate en suivant le parcours de la tableau 2 alors qu'il n'apparait pas dans w.

L'article [3] explique plusieurs méthodes pour améliorer la qualité musicale de la sortie de l'oracle en modifiant les possibilités de parcours :

— **Ne pas utiliser les liens facteurs :** ils entrainent des recombinaisons

Lettres	Flèches parcourues
b	$0 \to 2$
a	$2 \to 5$
c	$5 \to 6$
b	$6 \to 7$

Tableau 1 – Parcours de l'oracle des facteurs construit à partir du mot $w =$ *abcbacbaba* (Fig. 1) pour reconnaitre le facteur *bacb*.

Lettres	Flèches parcourues
a	$0 \to 1$
b	$1 \to 2$
a	$2 \to 5$

Tableau 2 – Parcours de l'oracle des facteurs construit à partir du mot $w =$ *abcbacbaba* (Fig. 1) amenant au faux-positif *aba*.

faisant perdre de la cohérence musicale. Par exemple, par des incohérences rythmiques, ou par des sauts avec un contexte commun très court après l'utilisation d'un lien suffixe.

— **Utiliser les liens suffixes :** ne pas utiliser les liens facteurs empêche d'avancer autrement que linéairement dans l'oracle. On utilise alors les liens suffixes, initialement prévus uniquement pour la construction de l'oracle, mais portant des liens entre les zones de l'oracle ayant un contexte commun, et donc par conséquent des liens avec plus de cohérence musicale. L'utilisation de ces liens suffixes seule n'est pas suffisante, car elle entrainerait une génération qui passe son temps à aller en arrière et qui resterait coincée au début de l'oracle. Pour répondre à ce problème, on utilise les liens suffixes inverses c'est-à-dire les liens suffixes dont la direction est inversée.

— **Éviter les premières occurences :** par construction, les liens suffixes se dirigent vers les premières occurences des facteurs ce qui entraine un risque de monotonie dans la génération musicale. Pour contrer ce problème, à partir d'un état on prend en compte :

1. le lien suffixe qui sort de cet état,
2. les liens suffixes inverses qui sortent de cet état,
3. le lien suffixe et les liens suffixes inverses sortant de la cible du lien suffixe précédent,
4. on réitérère l'étape 3 jusqu'à l'état 0 ou jusqu'à un certain critère d'arrêt.

Ainsi on obtient un certain nombre de candidats avec leurs longueurs de contexte commun respectives. Le choix du candidat est effectué aléatoirement en favorisant ceux avec un long contexte commun mais en laissant une chance à ceux ayant un contexte commun plus court. Avec ces trois premières méthodes, on peut alors imaginer le parcours suivant : *abacbabacb* (cf. Tableau 3)

— **Mettre en place un facteur de continuité :** il correspond au nombre minimal et maximal de notes successives jouées avant d'effectuer un saut.

Lettres	Flèches parcourues
a	$0 \rightarrow 1$
b	$1 \rightarrow 2$
a	$2 \dashrightarrow 4 \rightarrow 5$
c	$5 \rightarrow 6$
b	$6 \rightarrow 7$
a	$7 \rightarrow 8$
b	$8 \rightarrow 9$
a	$9 \dashrightarrow 2 \dashrightarrow 4 \rightarrow 5$
c	$5 \rightarrow 6$
b	$6 \dashrightarrow 3 \rightarrow 4$

Tableau 3 – Exemple de parcours de l'oracle des facteurs construit à partir du mot $w = abcbacbaba$ (Fig. 1) prenant en compte les méthodes proposées dans [3].

Il est réglable.

— **Chercher le bon endroit pour sauter :** en fixant un facteur de continuité trop strict, on risque de faire des sauts à des endroits où le contexte commun est faible. On prend alors en compte plutôt une zone autour de l'état où le saut devrait avoir lieu. Si le facteur de continuité est N, on effectue la recherche entre $N \pm N/5$. Encore une fois, le choix du candidat est effectué aléatoirement en favorisant ceux avec un long contexte commun mais en laissant une chance à ceux ayant un contexte commun plus court.

— **Répétition et tabou :** Les boucles peuvent être un véritable problème, surtout lorsqu'on optimise les sauts. On met alors une liste tabou d'une taille M correspondant aux M dernières cibles de saut. Il est alors interdit d'effectuer un saut vers une cible étant dans la liste. On met également dans la liste le voisinage des cibles afin d'éviter les fausses boucles.

OMax permet, grâce à la structure d'oracle des facteurs, d'effectuer un apprentissage en temps réel du langage musical d'un musicien et de générer une improvisation par un simple parcours d'automate. Cependant, cette apprentissage est unidimensionnel, ne prenant en compte qu'une seule caractéristique (hauteur de note, spectre...) pour construire l'oracle des facteurs. De plus, OMax ne possède pas d'écoute réactive de son environnement et ne peut donc pas s'adapter à ce qui est joué en temps réel.

3.1.2 ImproTeK

L'objectif d'ImproTeK [22, 23] est d'intégrer des contrôles rythmiques et harmoniques afin que le programme soit capable d'improviser sur un scénario (par exemple une grille d'accords) donné au préalable. OMax fait de chaque élément musical isolé un état dans l'oracle. Dans ImproTek, l'improvisation s'effectue dans une structure symbolique définie au début de chaque session. De plus, ImproTek intègre un cadre rythmique en définissant une pulsation comme unité élémentaire pour l'acquisition, la restitution et la génération.

Dans ImproTeK, les états de l'oracle sont donc des tranches musicales dont la durée est définie par la pulsation du morceau et qui contiennent les événements

ayant lieu entre deux pulsations (fragments mélodiques et accords). Le parcours se fait comme dans un automate pour lequel les transitions sont étiquetées par les accords de la grille. Un paramètre de continuité est également calculé et correspond à la longueur des segments recopiés. On peut fixer une longueur maximum. À chaque étape, on lit un accord dans la grille. Si la valeur courante du paramètre de continuité est inférieure au maximum imposé, la recherche est effectuée en suivant trois modes hiérarchisés :

— *Mode 'continuité'* : S'il existe une transition avec la bonne étiquette, on la suit, on met à jour la position de la pulsation courante dans l'oracle, et on retourne l'élément mélodique associé. Sinon, on passe en mode 'suffixe'.

— *Mode 'suffixe'* : On suit le lien suffixe correspondant au plus long suffixe répété pour atteindre ensuite une étiquette correspondante, si des liens suffixes avec la bonne étiquette sont trouvés, on met à jour la position de la pulsation courante dans l'oracle et on retourne l'élément mélodique associé. Sinon, on passe en mode 'vide'.

— *Mode 'vide'* : La recherche est effectuée sans prendre en compte le contexte. On cherche l'étiquette dans tout l'oracle, et une transposition peut avoir lieu si nécessaire.

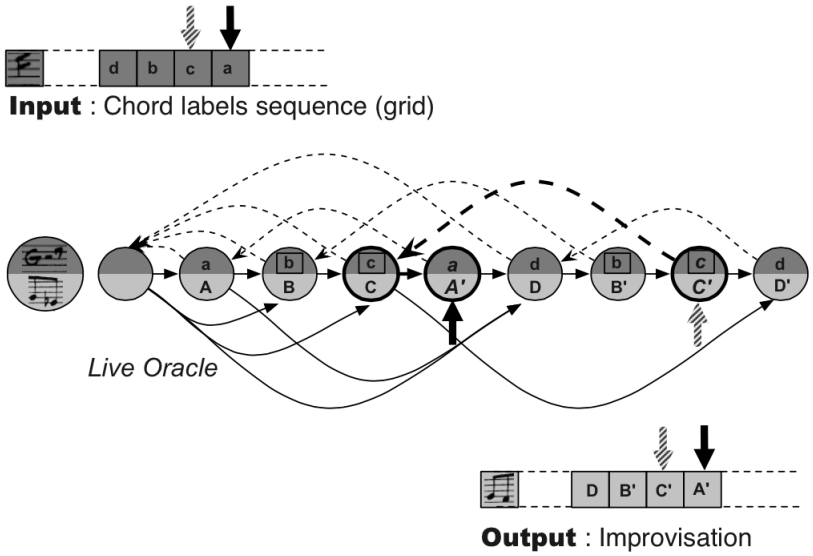

FIGURE 2 – Navigation dans l'Oracle par ImproTek [22].

Je reprends dans la Fig. 2 l'exemple proposé dans [22] pour plus de clarté. Il illustre une étape de la navigation. Les accords d, b, et c ont été cherchés et trouvés dans l'oracle, et la concaténation des fragments musicaux associés D, B', et C' forme la phrase en construction. La position courante dans l'oracle est l'avant-dernier état (c/C') et l'accord suivant lu dans la grille donnée en entrée est a. Aucune transition pointant sur le même accord ne peut être trouvée, on effectue donc la recherche en suivant les liens suffixes. Le lien suffixe partant de l'état courant pointe, par définition, sur l'état final du suffixe de dbc répété le plus à gauche (bc), soit l'état (c/C). Une transition correspondant à l'accord

a recherché est trouvée dans cet état. Le lien suffixe est alors suivi pour pouvoir atteindre la nouvelle pulsation (a/A'), et A' est concaténé à la phrase en construction.

Improtek permet de générer des improvisations guidées par un scénario. De cette manière, il est capable de mélanger plusieurs dimensions musicales. Par exemple, il est capable d'apprendre comment improviser des mélodies sur une grille d'accords. Cependant, l'apprentissage reste en soi unidimensionnel, la construction de l'oracle étant seulement effectuée par rapport à un scénario unidimensionnel. De plus, le scénario doit être connu au préalable, et il n'est pas possible de l'apprendre en temps réel.

3.1.3 SoMax

L'objectif de SoMax [8] est de donner des capacités d'écoute à OMax pour le rendre plus réactif. L'idée est de guider la navigation en prenant en compte ce que l'improvisateur joue en temps réel. Au lieu de choisir un état en se basant uniquement sur la taille du contexte commun, les états sont également évalués selon l'environnement musical présent. SoMax se base sur un principe d'activation de mémoire. À chaque instant, SoMax essaie de comprendre l'environnement musical dans lequel il est. Cela prend en compte un contexte d'apprentissage qui correspond à la représentation de la mémoire, et un contexte de jeu qui correspond à ce qui est joué à cet instant. Le contexte de jeu comprend à la fois ce qui est joué par les musiciens et ce qui est joué par SoMax.

Contrairement à OMax, la mémoire n'est pas représentée par un oracle des facteurs. Dans SoMax la mémoire est indexée par des n-grammes. Un n-gramme est une sous-séquence de n éléments consécutifs d'une séquence donnée. Cette structure est beaucoup utilisée dans le domaine du traitement automatique des langues. Ces n-grammes permettent de prédire quel sera le symbole x_i sachant les symboles $x_{i-(n-1)}, ..., x_{i-1}$. Autrement dit, ils définissent les probabilités $P(x_i|x_{i-(n-1)}, ..., x_{i-1})$. Dans le cadre de SoMax, ils permettent de prédire quelle note jouer en fonction des $n-1$ notes précédentes, et donc d'activer la mémoire en conséquence.

On présente ici les principaux ajouts de SoMax par rapport à OMax :

— **Segmentation et representation en séquence :** L'improvisation est segmentée en phrases par une détection de silence. Cela permet de définir les endroits où peut commencer et finir une improvisation. De plus, la durée des pauses peut être modifiée. Les pauses peuvent également être évitées ou favorisées lors du parcours de l'oracle pour satisfaire le discours musical. Les phrases sont analysées par une représentation basée sur les intervalles entre notes successives ce qui permet leur transposition. Ceci permet une augmentation conséquente du nombre de recombinaisons des différentes phrases, et permet aux phrases d'être jouées dans des contextes différents.

— **Synchronisation et conservation de pulsation :** Un système de suivi de pulsation est utilisé pour extraire en temps-réel la pulsation locale implicite du musicien afin de pouvoir se synchroniser. Lors de l'apprentissage, chaque élement musical (chaque état) est annoté avec sa position par rapport à la pulsation. Ainsi, lors de la génération, des positions

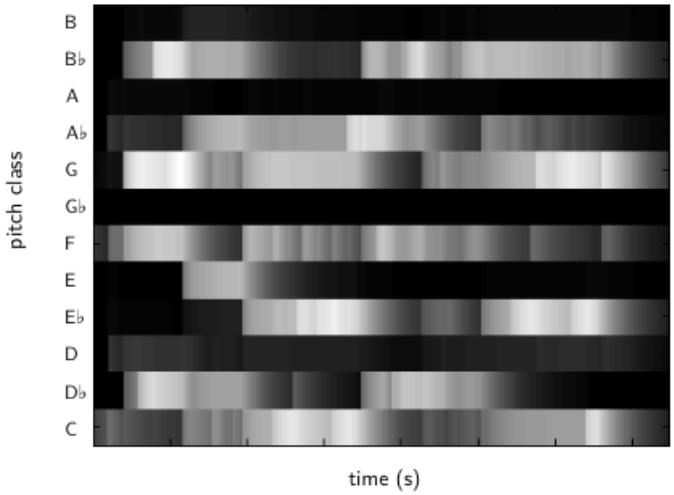

FIGURE 3 – Exemple de contexte harmonique d'un accompagnement en fonction du temps [8].

similaires seront favorisées. Ce principe fonctionne à plusieurs échelles ; il permet une synchronisation mais aussi de conserver la sensation de pulsation et l'éventuel 'swing' du musicien.

— **Contexte harmonique :** L'idée est d'être capable d'effectuer une analyse horizontale (mélodique) mais aussi verticale (harmonique) de l'information musicale. Chaque partie peut-être aussi complexe que voulue : solo, accords, clusters, polyphonies complexes. L'accompagnement est utilisé pour étiqueter le solo avec des informations harmoniques. Il est conservé dans un chromagramme. À noter qu'une note a une contribution qui persiste légèrement dans le temps en disparaissant progressivement. Cela permet de capturer les couleurs harmoniques locales. La Figure 3 montre l'évolution du contexte harmonique d'un accompagnement au cours du temps. Lors de la génération, ces étiquettes harmoniques sont utilisées pour diriger la navigation.

SoMax permet d'ajouter un principe de réactivité à OMax par une écoute active de son environnement. Comme Improtek, il est capable de mélanger plusieurs dimensions musicales, en focalisant son attention sur une dimension et en apprenant une autre. Par exemple, écouter une mélodie pour l'accompagner avec des accords. Cependant, l'apprentissage reste encore une fois unidimensionnel.

3.2 Modèles probabilistes

Lors de ce stage nous nous sommes intéressés à l'intégration de modèles probabilistes pour l'improvisation. Ce type d'approche a déjà été étudiée, notamment par Daniel Conklin [11] et par François Pachet *et al.* [25, 21], avec l'utilisation de modèles de Markov cachés. Nous présentons ici les principes de modélisation du langage ainsi que les méthodes d'interpolation de sous-modèles

en se basant principalement sur les travaux de Raczynski *et al.* [27, 28].

3.2.1 Modélisation du langage et n-grammes

L'objectif de la modélisation probabiliste du langage est de représenter sous forme de probabilité $P(s)$ la fréquence d'occurence d'une phrase s dans un langage [31]. Par exemple, considérons le francais. On peut imaginer que

$$P(\text{Bonjour}) \approx 0.01$$

en supposant qu'une phrase sur cent prononcée en français est la phrase «Bonjour». Cependant, on peut également imaginer que

$$P(\text{vous faites sirop de vingt-et-un}) \approx 0$$

car bien que cette phrase soit syntaxiquement correcte, il semblerait étrange que quelqu'un l'utilise.
Les modèles de langage probabilistes nécessitent d'être entrainés sur un corpus d'apprentissage. Ils ne sont donc pas aussi complet que les grammaires pour la reconnaissance d'un langage, le corpus ne pouvant parfois pas représenter l'ensemble du langage. Cependant, ils permettent d'obtenir de bons résultats lorsqu'il est impossible ou difficile de créer une grammaire définissant le langage que l'on souhaite modéliser. De plus, ils permettent d'évaluer les probabilités d'apparition de chaque phrase ; une grammaire non probabiliste n'ayant pas moyen de savoir si la phrase «Bonjour» est plus courante que la phrase «vous faites sirop de vingt-et-un».

Les n-grammes sont parmi les modèles de langage les plus courammment utilisés. Les modèles de n-grammes considèrent que la probabilité d'apparition d'un mot dépend seulement des $n-1$ mots qui le précèdent. Si on considère la phrase $s = w_1...w_N$, on suppose que :

$$P(w_i|w_1...w_{i-1}) \simeq P(w_i|w_{i-n+1}...w_{i-1}) \tag{1}$$

On peut alors évaluer la probabilité de la phrase complète comme :

$$P(s) = \prod_{i=1}^{N} P(w_i|w_1...w_{i-1}) \simeq \prod_{i=1}^{N} P(w_i|w_{i-n+1}...w_{i-1}) \tag{2}$$

La modélisation du langage est utilisée pour des tâches de reconnaissance automatique de la parole, de traduction automatique, de recherche d'information, *etc.* Elle est aussi utilisée en musique pour des tâches de génération comme peut, par exemple, le montrer le logiciel SoMax ou Orpheus [13], logiciel de composition automatique crée par Fukayama *et al.* utilisant la prosodie des paroles japonaises comme fil directeur.

3.2.2 Lissage des modèles

Lors d'un apprentissage sur un corpus, il est courant que les n-grammes observés ne couvrent pas l'ensemble des n-grammes qui peuvent être observés au moment du test. Ceci est vrai en particulier lorsque les corpus sont de taille restreinte. Ceci entraine des probabilités d'événements nulles, et peut empêcher

la prise en considération de certains éléments pouvant être possibles bien que non observés lors de l'apprentissage. Le but du lissage est de corriger l'estimation des probabilités (pas forcément nulles) estimées à partir d'un nombre réduit d'occurences.
Il existe deux principales méthodes de lissage [9].

Le lissage additif : L'idée du lissage additif est de considérer que chaque élement apparait δ fois plus qu'il n'apparait vraiment, avec typiquement $0 < \delta \leq 1$. À partir de l'estimation des probabilités au sens du maximum de vraisemblance (MV) $P_{\text{MV}}(X|Y) = \frac{c(X,Y)}{\sum_{X'} c(X',Y)}$, avec c la fonction comptant le nombre d'occurences d'un élément dans l'ensemble d'apprentissage, le lissage additif donne :

$$P_{\text{add}}(X|Y) = \frac{\delta + c(X,Y)}{\sum_{X'} \delta + c(X',Y)} \tag{3}$$

Par exemple, considérons le mot : $w = acacbc$. Sans lissage nous avons les probabilités suivantes : $P(w_i = c|w_{i-1} = a) = 1$, $P(w_i = c|w_{i-1} = b) = 1$, et $P(w_i = c|w_{i-1} = c) = 0$. En effectuant un lissage additif avec $\delta = 1$, les probabilités deviennent : $P(w_i = c|w_{i-1} = a) = 3/5$, $P(w_i = c|w_{i-1} = b) = 1/2$, et $P(w_i = c|w_{i-1} = c) = 1/5$.
Ce type de lissage est typiquement utile lorsque l'apprentissage est effectué sur un corpus petit et que l'ordre n des modèles est faible (1 ou 2). Il permet de remédier aux erreurs de reconnaissance dues à des probabilités d'événement nulles.

Le lissage par repli : Le lissage par repli est une généralisation du lissage additif. L'idée sous-jacente est d'utiliser des informations de modèles d'ordre inférieur afin de pallier les problèmes de surapprentissage d'un modèle trop spécifique pour certains contextes.
Considérons comme exemple le mot : $w = aaaabaaaaaaacd$. Si nous nous intéressons au modèle de bigramme sur ce mot, nous avons : $P(w_i = a|w_{i-1} = c) = 0$ et $P(w_i = b|w_{i-1} = c) = 0$. Si nous effectuons un lissage additif sur ce bigramme, nous obtiendrions $P(w_i = a|w_{i-1} = c) = P(w_i = b|w_{i-1} = c)$. Or intuitivement, en regardant la proportion de a et de b dans le mot, nous aurions tendance à vouloir : $P(w_i = a|w_{i-1} = c) > P(w_i = b|w_{i-1} = c)$.
Nous pouvons alors effectuer une interpolation du modèle de bigramme et du modèle d'unigramme :

$$P_{\text{interp}}(w_i|w_{i-1}) = \lambda P_{\text{MV}}(w_i|w_{i-1}) + (1-\lambda)P_{\text{add}}(w_i) \tag{4}$$

On peut alors généraliser ceci à tout type de modèle :

$$P_{\text{interp}}(X|Y) = \lambda_Y P_{\text{MV}}(X|Y) + (1-\lambda_Y)P(X|Z) \tag{5}$$

où $Z \subset Y$ est un sous-ensemble des variables Y. Dans le cadre des n-grammes, on peut avoir : $Y = w_{i-n+1\ldots i-1}$ et $Z = w_{i-n+2\ldots i-1}$.

3.2.3 Interpolation de sous-modèles

Dans l'article [27], Raczynski *et al.* proposent un modèle permettant d'effectuer l'harmonisation automatique de mélodies. Ce modèle permet de prédire la

progression d'accords en fonction de plusieurs variables :

$$P(C_t|X_{1:t}) \tag{6}$$

où C_t représente l'accord joué à la trame t, et $X_{1:t}$ correspond à un ensemble de variables musicales (par exemple : notes de la mélodie, accords, tonalités, vélocités,...) sur l'ensemble des trames de 1 à t. Ceci permet de représenter des corrélations entre plusieurs dimensions musicales.
Cependant, ce modèle est d'une dimension trop élevée et ne peut donc pas être utilisé en pratique. Pour remédier à ce problème on effectue alors une interpolation de différents sous-modèles plus simples ne prennant en compte qu'une partie des variables $A_{i,t} \subset X_{1:t}$ [28]. Par exemple, les deux accords précédents $A_{1,t} = (C_{t-1}, C_{t-2})$ et la tonalité à l'instant t $A_{2,t} = T_t$.
Cette interpolation peut-être linéaire [14] :

$$P(C_t|X_{1:t}) = \sum_{i=1}^{I} \lambda_i P_i(C_t|A_{i,t}) \tag{7}$$

avec λ_i des coefficients d'interpolation tels que $\sum_{i=1}^{I} \lambda_i = 1$, $\lambda_i \geq 0$ pour tout $i = 1, ..., I$, et I le nombre de sous-modèles.
L'interpolation log-linéaire [15] a aussi été étudiée :

$$P(C_t|X_{1:t}) = Z^{-1} \prod_{i=1}^{I} P_i(C_t|A_{i,t})^{\gamma_i} \tag{8}$$

avec γ_i les coefficients d'interpolation tels que $\gamma_i \geq 0$ pour tout $i = 1, ..., I$, et Z un facteur de normalisation dépendant des $A_{i,t}$ défini par :

$$Z = \sum_{C_t} \prod_{i=1}^{I} P_i(C_t|A_{i,t})^{\gamma_i} \tag{9}$$

Les résultats obtenus dans [27] montrent que l'interpolation de sous-modèle peut permettre d'obtenir des résultats plus précis pour l'harmonisation automatique de mélodies.

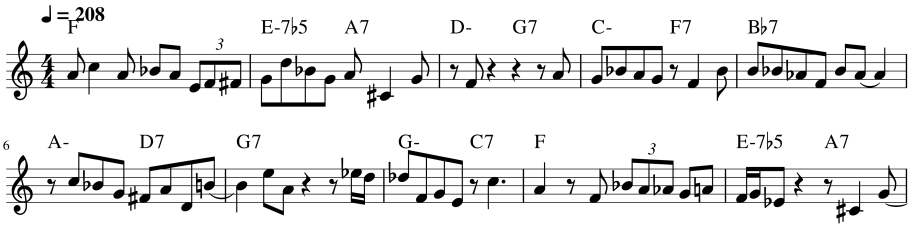

FIGURE 4 – Premières mesures du thème Confirmation de Charlie Parker ; exemple typique du style bebop.

4 Utilisation de modèles probabilistes pour l'improvisation automatique

4.1 Présentation des tâches et du corpus

La méthode que nous allons étudier s'approche de celle de SoMax dans le sens où elle se base sur des modèles de n-grammes. Contrairement à SoMax cependant, nous utiliserons des modèles combinant différentes dimensions musicales. Nous allons appliquer les méthodes d'interpolation de sous-modèles pour effectuer deux tâches importantes dans le contexte de l'improvisation automatique afin de valider si de telles méthodes peuvent fournir des résultats intéressants permettant de prendre en considération de multiples dimensions musicales lors de la réalisation d'une improvisation automatique. Premièrement, nous allons nous intéresser à l'harmonisation automatique. L'objectif est ici de prédire quel accord jouer en fonction du contexte en prenant en compte à la fois l'harmonie et la mélodie. Ceci peut être utile pour des tâches d'accompagnement automatique. Deuxièmement, nous nous intéresserons à la génération de mélodie. L'objectif est ici de prédire quelle note jouer en fonction du contexte pour créer une mélodie, en prenant en compte encore une fois l'harmonie et la mélodie.

Afin de tester ces méthodes, nous allons utiliser comme corpus l'Omnibook de Charlie Parker [26]. Il contient cinquante thèmes et improvisations joués par Charlie Parker, principalement dans le style bebop ; type de jazz avec un tempo très rapide, et des grilles harmoniques avec beaucoup de changements d'accords. Comme on peut le voir sur le thème Confirmation, écrit par Charlie Parker, dans la Fig. 4, le tempo est très élevé (certains morceaux atteignent 320 à la noire), et il est fréquent d'avoir deux accords par mesure. Dans l'Omnibook, les accords sont organisés en cinq classes : majeur, mineur, dominante, demi-diminué, et diminué. La durée moyenne d'un morceau est d'environ 64 mesures. Le nombre total de temps est 14328, pour un total de 21831 notes.

Ce corpus possède plusieurs avantages pour nos applications par rapport au corpus utilisé dans [27] pour nos applications. Tout d'abord, il s'agit d'un réel corpus d'improvisation, avec des transcriptions précises des solos d'une grande figure de l'histoire du jazz, au lieu d'un corpus de musique écrite. De plus, il est très représentatif d'un style en particulier, ce qui permet d'avoir un corpus avec une grande cohérence. Cependant, ce corpus est d'une taille très limitée

en comparaison au corpus utilisé dans [27] avec seulement 50 morceaux contre 2000.
Nous avions avant ça effectué quelques tests sur la Weimar Jazz Database (WJazzD), créée dans le cadre du projet Jazzomat [1]. Ce corpus comporte 299 transcriptions de solo de jazz dans des styles variés (swing, bebop, cool jazz, free jazz...). Cependant, ces transcriptions ne comportent que l'information mélodique et le thème n'apparait pas. De plus, il s'agit de transcriptions automatiques qui possèdent donc des imprécisions, notament sur le plan rythmique. Nous avons corrigé les imprécisions et ajouté les informations harmoniques à la main sur 15 solos afin d'effectuer des tests. Cependant, le corpus établi était trop léger et trop varié, et nous n'avons pas obtenus de résultats satisfaisant avec celui ci. Nous avons alors décidé de nous tourner vers l'Omnibook.

Nous avons retranscrit l'Omnibook à la main sur le logiciel MuseScore (éditeur de partition). Les données ont ensuite été transformé à l'aide de MuseScore au format musicXML. Nous avons ensuite écrit un *parser* en python pour pouvoir utiliser ces données dans notre code.
Nous avons séparé ce corpus en trois :

— un corpus d'apprentissage composé de 40 morceaux,
— un corpus de validation composé de 5 morceaux,
— un corpus de test composé de 5 morceaux.

Le corpus d'apprentissage est utilisé pour apprendre les différentes probabilités des sous-modèles que nous allons utiliser. Le corpus de validation est utilisé pour optimiser les coefficients d'interpolation et de lissage. Et le corpus de test est utilisé pour évaluer les résultats. La métrique utilisée pour l'optimisation des coefficients et pour l'évaluation des résultats est l'entropie croisée définie dans l'équation (24) de la partie 4.5.

4.2 Représentations utilisées

Les notes sont représentées uniquement par leur hauteur sans prendre en considération leur octave. Les accords sont représentés par leur fondamental et par leur nature.
Les partitions sont divisées en trames. Nous utiliserons des trames allant d'une durée de un demi-temps jusqu'à une durée de quatre temps. La mélodie jouée sur une trame donnée correspond à la liste des notes dont l'attaque a lieu dans cette trame sans prendre en considération l'ordre temporel. Par exemple, dans la Fig. 5, nous avons effectué une segmentation par trames de deux temps. Dans la première trame, la mélodie est (ré,mi). Dans la seconde trame, la mélodie est (do,sol), la répétition du do n'apparait pas (les octaves ne sont pas considérées). Les mélodies de la troisième et de la quatrième trame sont vides.
De même, si plusieurs accords sont joués dans une seule trame, ils sont représentés par une liste ne prennant pas en considération leur ordre.

Dans le début de la partie 4.5.2, lorsque nous effectuons un apprentissage note par note, les trames n'ont pas une durée fixe. Elles ont la durée des notes prises une par une. Les mélodies sont alors constituées d'une seule note, et pour chaque trame l'accord considéré est le dernier accord joué.

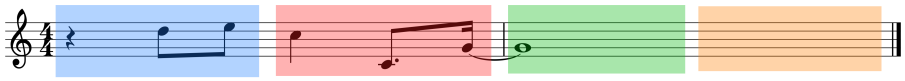

FIGURE 5 – Exemple de segmentation par trames de deux temps.

4.3 Sous-modèles

Pour la tâche d'harmonisation nous utilisons les deux sous-modèles suivants, correspondant aux deux élements semblant les plus importants pour cette tâche :

$$P_{C1}(C_t|X_{1:t}) = P(C_t|C_{t-1}) \tag{10}$$

$$P_{C2}(C_t|X_{1:t}) = P(C_t|M_t) \tag{11}$$

C_t correspond à l'accord joué à la trame t, et M_t correspond à la mélodie jouée à la trame t. Les mélodies sont représentées par l'ensemble des notes jouées lors du temps. Les rythmes ne sont pas pris en compte.

P_{C1} est un bigramme sur les accords. Il semble logique qu'un accord a de fortes chances d'être influencé par l'accord qui le précède. Afin de ne pas être influencé par la tonalité et par les problèmes de transposition, lors de l'apprentissage, nous allons considérer des transitions d'accords relatives en considérant toutes les transpositions possibles des morceaux observés. Ainsi, la probabilité de passer de G^7 à C^{Maj} sera identique à la probabilité de passer de A^7 à D^{Maj}.

P_{C2} permet de prendre en considération les relations entre harmonie et mélodie. De même que pour le bigramme, pour ne pas être influencé par les problèmes de transposition, nous considérons des relations relatives entre la mélodie et l'accord. Ainsi, la probabilité de jouer (C, D, E) sur un accord de C^{Maj} sera identique à la probabilité de jouer $(D, E, F\sharp)$ sur un accord de D^{Maj}.

Pour la tâche de génération de mélodie, nous utilisons les deux sous-modèles suivants :

$$P_{M1}(M_t|X_{1:t}) = P(M_t|M_{t-1}) \tag{12}$$

$$P_{M2}(M_t|X_{1:t}) = P(M_t|C_t) \tag{13}$$

De même que pour la tâche précédente, nous utilisons des intervalles relatifs entre fragments mélodiques successifs afin de pas être soumis aux problèmes de transposition.

4.4 Apprentissage

Nous effectuons une interpolation de ces sous-modèles avec une combinaison de lissage additif et de lissage par repli en utilisant respectivement $P(C_t)$ et $P(M_t)$ comme modèle d'ordre inférieur pour les tâches d'harmonisation et de génération de mélodie. Dans le cas de l'interpolation linéaire, on obtient alors respectivement pour les deux tâches

$$P(C_t|X_{1:t}) = \alpha_C P(C_t) + \beta_C U(C_t) + a_{C1} P(C_t|C_{t-1}) + a_{C2} P(C_t|M_t) \tag{14}$$

et

$$P(M_t|X_{1:t}) = \alpha_M P(M_t) + \beta_M U(M_t) + a_{M1} P(M_t|M_{t-1}) + a_{M2} P(M_t|C_t) \quad (15)$$

avec U la distribution uniforme,

$$\alpha_C + \beta_C + a_{C1} + a_{C2} = 1 \quad (16)$$

et

$$\alpha_M + \beta_M + a_{M1} + a_{M2} = 1 \quad (17)$$

Pour l'interpolation log-linéaire, on doit lisser les sous-modèles séparément, on obtient alors respectivement pour les deux tâches

$$P(C_t|X_{1:t}) = \frac{1}{Z_C} \prod_{i=1}^{2} (\gamma_{Ci} P_{Ci}(C_t|A_{i,t}) + \delta_{Ci} P(C_t) + \varepsilon_{Ci} U(C_t))^{b_{Ci}} \quad (18)$$

et

$$P(M_t|X_{1:t}) = \frac{1}{Z_M} \prod_{i=1}^{2} (\gamma_{Mi} P_{Mi}(M_t|A_{i,t}) + \delta_{Mi} P(M_t) + \varepsilon_{Mi} U(M_t))^{b_{Mi}} \quad (19)$$

avec pour tout i

$$\gamma_{Ci} + \delta_{Ci} + \varepsilon_{Ci} = 1 \quad (20)$$

et

$$\gamma_{Mi} + \delta_{Mi} + \varepsilon_{Mi} = 1 \quad (21)$$

Z_C et Z_M sont les constantes de normalisation définies par

$$Z_C = \sum_{C_t} \prod_{i=1}^{I} (\gamma_{Ci} P_{Ci} + \delta_{Ci} P(C_t) + \varepsilon_{Ci} U(C_t))^{b_{Ci}} \quad (22)$$

$$Z_M = \sum_{M_t} \prod_{i=1}^{I} (\gamma_{Mi} P_{Mi} + \delta_{Mi} P(M_t) + \varepsilon_{Mi} U(M_t))^{b_{Mi}} \quad (23)$$

L'apprentissage des coefficients est effectué par raffinements successifs afin d'obtenir un minimum d'entropie croisée. Tous les coefficients sont entrainés en même temps. Les valeurs des coefficients sont cherchées de 0 à 1 d'abord par pas de 0,1, puis par pas de 0.01 autour de l'optimum trouvé ($\pm$ 20%), et ainsi de suite. Ce type d'optimisation a pour risque de tomber dans un minimum local (même si cela semble improbable par la forme des résultats obtenus). D'autres heuristiques d'optimisation auraient pu être utilisées pour diminuer ce risque, comme par exemple un recuit simulé. Cependant, l'évolution des résultats en fonction des valeurs des coefficients tend à faire penser qu'il s'agit d'un problème convexe. Une étude théorique pourra être effectuée pour prouver la convexité de ce problème.

4.5 Résultats

L'évaluation des harmonies et mélodies générées peut être très subjective et demanderait d'être évaluée par des experts. Afin d'évaluer nos résultats nous allons utiliser la notion d'entropie croisée sur les données de test définie par

$$H(C) = -\frac{1}{T}\sum_{t=1}^{T}\log_2 P(C_t|X_{1:t}) \tag{24}$$

Ceci peut être interprété comme le nombre moyen de bits nécessaire pour encoder un seul accord ou fragment mélodique (au sens du codage de Shannon), et représente l'incompréhension du système. Une faible valeur d'entropie croisée indique alors un meilleur pouvoir de prédiction.

4.5.1 Tâche d'harmonisation

Concernant la tâche d'harmonisation, pour l'interpolation linéaire, nous obtenons les résultats du tableau 4. La première ligne du tableau correspond aux résultats optimaux obtenus lors de l'expérience.

On se rend compte que, pour cette tâche d'harmonisation, l'interpolation de sous-modèles n'est pas utile, et ceci pour des longueurs de trame allant d'un demi-temps jusqu'à quatre temps. On obtient en effet des résultats comparables en considérant uniquement le bigramme sur les accords. Ceci peut être expliqué par la nature du corpus. En effet, le bebop est une musique basée sur des grilles d'accords prédéfinies qui se répètent. De plus, beaucoup de ces grilles sont identiques ; en effet une grande partie du répertoire bebop est basé sur des grilles de blues ou des anatoles [17]. Cet aspect répétitif peut alors expliquer pourquoi l'utilisation du bigramme seul est suffisante pour la génération d'harmonisation dans ce style. De plus, on peut remarquer que le modèle mélodie/accord est tellement peu informatif qu'il ne contribue pas au modèle global quand utilisé seul. Nous obtenons les mêmes résultats pour une interpolation log-linéaire.

Dans [27], l'interpolation de sous-modèles permet d'obtenir un meilleur pouvoir de prédiction. Ceci peut s'expliquer par le fait que le corpus utilisé est plus grand, plus varié et possède un plus grand nombre d'accords (351 contre 60). On se rend alors compte de l'importance du corpus. Sur un corpus restreint avec une forte cohérence où l'harmonie progresse suivant une grille, un modèle de n-grammes possède un pouvoir de prédiction très supérieur à celui du modèle mélodie/accords et il n'est pas nécessaire d'effectuer une interpolation de sous-modèles, alors que sur un corpus très varié la mélodie peut apporter de l'information supplémentaire et l'interpolation de sous-modèles fournit de meilleurs résultats.

4.5.2 Génération de mélodie

Concernant la génération de mélodie, pour l'interpolation linéaire, nous avons tout d'abord essayé d'effectuer l'apprentissage note par note au lieu de prendre des trames mélodiques. Nous obtenous les résultats du tableau 5. La première ligne de la table correspond aux résultats optimaux obtenus lors de l'expérience.

	trame = 1/2 temps					trame = 1 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(C)$	a_{M1}	a_{M2}	α_M	β_M	$H(C)$
B+M	**0,999**	**0**	**0**	**0,001**	**1,01964587877**	**0,998**	**0**	**0**	**0,002**	**1,69881563208**
B	0,999	0	0	0,001	1,01964587877	0,998	0	0	0,002	1,69881563208
M	0	0	0.957	0.043	5,14039347682	0	0	0.957	0.043	5,13920330587
unigramme seul	0	0	0.957	0.043	5,14039347682	0	0	0.957	0.043	5,13920330587

	trame = 2 temps					trame = 4 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(C)$	a_{M1}	a_{M2}	α_M	β_M	$H(C)$
B+M	**0,997**	**0**	**0**	**0,003**	**2,59621828289**	**0,999**	**0**	**0**	**0,001**	**2,85689366181**
B	0,997	0	0	0,003	2,59621828289	0,999	0	0	0,001	2,85689366181
M	0	0	0.958	0.042	5,13670667932	0	0,915	0	0.085	5,11851992508
unigramme seul	0	0	0.958	0.042	5,13670667932	0	0	0.947	0.053	5,13121020315

Tableau 4 – Résultats d'entropie croisée (bits/trame) pour la tâche d'harmonisation avec interpolation linéaire. Nous obtenons les résultats pour l'interpolation des sous-modèles de bigramme et mélodie/accord (B+M), puis pour le bigramme seul (B), puis pour le modèle mélodie/accord (M), puis avec unigramme lissé seul.

	coefficients				Entropie croisée
	a_{M1}	a_{M2}	α_M	β_M	$H(M)$
B+M	**0,526**	**0,474**	**0**	**0**	**3,23070449427**
B	0,801	0	0,199	0	3,28532263409
M	0	0,756	0,244	0	3,30013321829
unigramme seul	0	0	1	0	3,4672916296

Tableau 5 – Résultats d'entropie croisée (bits/note) pour la tâche de génération de mélodie note à note avec interpolation linéaire. Nous obtenons les résultats pour l'interpolation des sous-modèles de bigramme et mélodie/accord (B+M), puis pour le bigramme seul (B), puis pour le modèle mélodie/accord (M), puis avec lissage seul.

Pour cette tâche, on se rend compte que la combinaison des deux sous-modèles permet d'obtenir de meilleurs résultats que l'utilisation d'un seul modèle. Utiliser à la fois l'information harmonique et mélodique permet d'avoir un meilleur pouvoir de prédiction pour la mélodie.
On peut noter que nous obtenons un résultat optimal lorsque nous n'effectuons aucun lissage. Ceci s'explique par le fait que tous les bigrammes de notes, et toutes les combinaisons note / accord ont été observées pendant l'apprentissage. Les notes étant représentées uniquement par leur hauteur et sans considération de l'octave. L'ensemble des possibilités (12 notes et 60 accords) est alors suffisament restreint pour ne pas nécessiter de lissage.

Nous avons ensuite réalisé des tests en extrayant les mélodies par trames, en faisant varier la longueur de trame de un demi-temps à quatre temps. Nous obtenons les résultats du tableau 6. On se rend compte que pour toutes les longueurs de trame, une combinaison des deux sous-modèles permet d'obtenir de meilleurs résultats que l'utilisation de chaque modèle séparément. On se rend compte également que le choix de la longueur de trame semble être un élément important pour l'apprentissage. En effet, nous obtenons des résultats très différents en fonction de ce paramètre. Le modèle de bigramme a une importance plus grande pour des longueurs de trame courtes, alors que le modèle mettant en relation mélodie et harmonie prend le dessus pour des longueurs de trame plus élevées. Ceci peut s'expliquer par le fait que les accords changent en moyenne une fois toutes les mesures, et donc que les trames de deux ou quatre temps caractérisent mieux l'évolution harmonique, alors que les trames d'un demi-temps ou d'un temps sont plus adaptées concernant l'évolution mélodique. À noter que l'on ne peut pas comparer les scores d'entropies croisées pour des trames de longueur différentes, le nombre de trames T influant dans le calcul de l'entropie croisée.

Nous avons également testé cette tâche en remplaçant le modèle $P_{M2} = (M_t|C_t)$ par le modèle $P_{M3} = (M_t|C_{t-1})$. Ce modèle correspond à une situation où l'improvisateur n'a aucune information sur les accords qui vont être joués et doit se contenter de baser son improvisation uniquement sur ce qu'il a entendu précédement. Ceci correspond aux systèmes sans écoute active. Ce modèle n'a pas vraiment de sens pour une génération de mélodie note à note, nous avons

	trame = 1/2 temps					trame = 1 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(M)$	a_{M1}	a_{M2}	α_M	β_M	$H(M)$
B+M	**0.693**	**0.031**	**0.276**	**0**	**2.95105059572**	**0.582**	**0.129**	**0.289**	**0**	**4.54280181704**
B	0.714	0	0.286	0	2.95554492413	0.672	0	0.328	0	4.57222521659
M	0	0.623	0.377	0	3.22436560353	0	0.639	0.361	0	4.88052016088
unigramme seul	0	0	1	0	3.71886700325	0	0	0.998	0.002	5.85758744742

	trame = 2 temps					trame = 4 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(M)$	a_{M1}	a_{M2}	α_M	β_M	$H(M)$
B+M	**0.187**	**0.508**	**0.303**	**0.002**	**6.82369543295**	**0.027**	**0.372**	**0.553**	**0.048**	**9.84626856472**
B	0.392	0	0.602	0.006	7.38202726006	0.082	0	0.762	0.156	10.4997806044
M	0	0.671	0.327	0.002	6.93710213698	0	0.390	0.557	0.053	9.86782433646
unigramme seul	0	0	0.992	0.008	8.39624755588	0	0	0.818	0.182	10.7126980855

Tableau 6 – Résultats d'entropie croisée (bits/trame) pour la tâche de génération de mélodie par trame avec interpolation linéaire. Nous obtenons les résultats pour l'interpolation des sous-modèles de bigramme et mélodie/accord (B+M), puis pour le bigramme seul (B), puis pour le modèle mélodie/accord (M), puis avec lissage seul.

alors effectué les tests pour des mélodies extraites par trames, en faisant varier la longueur de trame de un demi-temps à quatre temps. Nous obtenons les résultats du tableau 7.
Nous obtenons des résultats similaires qu'avec le modèle P_{M2}. La combinaison des deux sous-modèles permet d'obtenir de meilleurs résultats que l'utilisation de chaque modèle séparément. Les scores d'entropies croisées pour des trames de même longueur sont légèrement supérieurs avec le modèle P_{M3} qu'avec le modèle P_{M2} avec une différence s'amplifiant lorsque la longueur de trame grandit. Ceci s'explique par le fait qu'un bon improvisateur réagit rapidement à son environnement sonore. Lorsque la trame est trop longue, C_{t-1} caractérise moins bien ce qui est joué par l'improvisateur lors de la trame t.

De manière générale, ces résultats sont intéressant, l'interpolation des sous-modèles apportant un meilleur pouvoir de prédiction au modèle global dans tous les cas. Cependant, les améliorations des scores d'entropies semblent assez faibles. Il faut donc se demander si cette amélioration est suffisante. Ceci nécessiterait une écoute des résultats générés par des experts. Si l'amélioration est jugée suffisante par les experts, alors l'entropie croisée ne rend pas forcément bien compte de la qualité perçue. Ceci pourrait être justifié par le fait que l'entropie croisée mesure la capacité de reproduction d'un système, et que l'improvisation n'est pas une pratique basée uniquement sur la reproduction, mais sur la variété dans l'expressivité du musicien. Si l'amélioration n'est pas jugée suffisante par les experts, alors il serait nécessaire de combiner le système avec les principes de contexte commun utilisés dans OMax.

	trame = 1/2 temps					trame = 1 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(M)$	a_{M1}	a_{M2}	α_M	β_M	$H(M)$
B+P	**0.699**	**0.022**	**0.279**	**0**	**2.95184160927**	**0.617**	**0.083**	**0.300**	**0**	**4.55405504756**
B	0.714	0	0.286	0	2.95554492413	0.672	0	0.328	0	4.57222521659
P	0	0.612	0.388	0	3.23918839874	0	0.618	0.382	0	4.93439721379
unigramme seul	0	0	1	0	3.71886700325	0	0	0.998	0.002	5.85758744742

	trame = 2 temps					trame = 4 temps				
	a_{M1}	a_{M2}	α_M	β_M	$H(M)$	a_{M1}	a_{M2}	α_M	β_M	$H(M)$
B+P	**0.250**	**0.390**	**0.360**	**0**	**7.00279123948**	**0.034**	**0.342**	**0.572**	**0.052**	**10.0043803713**
B	0.392	0	0.602	0.006	7.38202726006	0.082	0	0.762	0.156	10.4997806044
P	0	0.596	0.404	0	7.1957637173	0	0.366	0.572	0.062	10.0342567312
unigramme seul	0	0	0.992	0.008	8.39624755588	0	0	0.818	0.182	10.7126980855

Tableau 7 – Résultats d'entropie croisée (bits/trame) pour la tâche de génération de mélodie par trame avec interpolation linéaire. Nous obtenons les résultats pour l'interpolation des sous-modèles de bigramme et mélodie/accord précédent (B+P), puis pour le bigramme seul (B), puis pour le modèle mélodie/accord précédent (P), puis avec lissage seul.

5 Conclusion et perspectives

Lors de ce stage, nous avons adapté et évalué l'interpolation de modèles probabilistes pour des tâches relatives à l'improvisation musicale. Nous avons montré que, dans certains cas, l'utilisation combinée d'informations provenant de différentes dimensions musicales permet d'avoir une meilleur compréhension du contexte et pourrait potentiellement fournir des résultats musicaux plus proches du style appris par la machine.

Nous avons constaté que la longueur de trame utilisé pour l'apprentissage est un élément important. Ceci avait déjà été remarqué par Raczynski *et al.* dans [27] pour l'harmonisation automatique, mais c'est également le cas pour la génération de mélodie. Il serait intéressant de pouvoir développer des moyens d'optimiser la longueur de trame. Une autre piste intéressante serait de combiner des modèles utilisant des tailles de trames différentes. Il serait aussi intéressant d'intégrer ce type de modèle aux logiciels d'improvisation existants afin de pouvoir effectuer de réelles évaluations par des experts.

Il serait également intéressant par la suite de multiplier le nombre de sous-modèles pris en considération, et travailler potentiellement sur de réels descripteurs audio [32]. Une autre piste pourrait également de réaliser des méthodes d'apprentissage permettant de créer une hiérarchie des différentes dimensions musicales afin de savoir si par exemple l'harmonie dirige la mélodie ou *vice versa*.

Une limite de ce système concerne le temps de calcul nécessaire pour l'apprentissage principalement dû à l'évaluation des coefficients d'interpolation et de lissage. Ceci n'est pas un problème pour une application où l'apprentissage est effectué avant la génération. Cependant, les logiciels actuels d'improvisation automatique fonctionnent en temps réel, il serait donc intéressant de développer des heuristiques permettant d'accélérer le processus d'apprentissage avec une adaptation contrôlée des paramètres de lissage au cours du temps. Ceci permettrait de pouvoir effectuer les processus d'apprentissage et de génération simultanément en temps réel.

Par la suite, il serait également intéressant de faire des comparaisons avec d'autres méthodes d'apprentissage automatique telles que des méthodes d'apprentissage par renforcement [33] pouvant permettre une meilleur adaptabilité du système, ou des méthodes basées sur des réseaux de neurones [30, 6]. Une autre tâche à explorer est l'apprentissage de forme multi-échelle [7]. L'objectif ici serait de modéliser des formes par blocs («aaba», «abca», couplet-refrain, ...), très présentes dans le jazz, la musique classique, et la musique pop, puis d'étendre ce modèle aux structures pouvant émerger dans l'improvisation libre.

De manière générale, ce stage fut enrichissant, et apporte des résultats intéressants. Je suis personnellement ravi de pouvoir continuer de travailler sur ce projet à la suite de ce stage, dans le cadre d'une thèse à Inria Nancy.

Références

[1] Jakob Abeßer, Klaud Frieler, Martin Pfleiderer, and Wold-Georg Zaddach. Introducing the jazzomat project - jazz solo analysis using music information retrival methods. In *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, pages 653–661, 2013.

[2] Cyril Allauzen, Maxime Crochemore, and Mathieu Raffinot. Factor oracle : A new structure for pattern matching. *SOFSEM'99, Theory and Practice of Informatics*, pages 291–306, 1999.

[3] Gérard Assayag and Georges Bloch. Navigating the oracle : A heuristic approach. In *Proceedings of the International Computer Music Conference*, pages 405–412, 2007.

[4] Gérard Assayag, Georges Bloch, and Marc Chemillier. OMax-OFon. In *Proceedings of the Sound and Music Computing Conference*, 2006.

[5] Gérard Assayag and Shlomo Dubnov. Using factor oracles for machine improvisation. *Soft Computing*, 8-9 :604–610, 2004.

[6] Greg Bickerman, Sam Bosley, Peter Swire, and Robert M. Keller. Learning to create jazz melodies using deep belief nets. In *Proceedings of the International Conference on Computational Creativity*, pages 228–236, 2010.

[7] Frédéric Bimbot, Gabriel Sargent, Emmanuel Deruty, Corentin Guichaoua, and Emmanuel Vincent. Semiotic description of music structure : An introduction to the Quaero/Metiss structural annotations. In *Proceedings of the AES 53rd International Conference on Semantic*, pages 32–43, 2014.

[8] Laurent Bonasse-Gahot. An update on the SoMax project. Technical report, IRCAM, 2014.

[9] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.

[10] Alain De Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4) :1917–1930, 2002.

[11] Darrell Conklin. Music generation from statistical models. In *Proceedings of the AISB Symp. on Artificial Intelligence and Creativity in the Arts and Sciences*, pages 30–35, 2003.

[12] Alexandre Donze, Sophie Libkind, Sanjit A. Seshia, and David Wessel. Control improvisation with application to music. Technical Report UCB/EECS-2013-183, EECS Department, University of California, Berkeley, November 2013.

[13] Satoru Fukayama, Kei Nakatsuma, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama. Automatic song composition from the lyrics exploiting prosody of Japanese language. In *Proceedings of the 7th Sound and Music Computing Conference*, pages 299–302, 2010.

[14] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*, pages 381–397, 1980.

[15] Dietrich Klakow. Log-linear interpolation of language models. In *Proceedings of the 5th International Conference on Spoken Language Processing*, pages 1695–1698, 1998.

[16] Arnaud Lefebvre and Thierry Lecroq. Computing repeated factors with a factor oracle. In *Proceedings of the 11th Australasian Workshop On Combinatorial Algorithms*, pages 145–158, 2000.

[17] Mark Levine. *The Jazz Theory Book*. Sher Music, 1995.

[18] Benjamin Lévy. Visualising OMax. Master's thesis, IRCAM, 2009.

[19] Benjamin Lévy, Georges Bloch, and Gérard Assayag. OMaxist dialectics : Capturing, visualizing and expanding improvisations. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 137–140, 2012.

[20] Fivos Maniatakos, Gerard Assayag, Frederic Bevilacqua, and Carlos Agon. On architecture and formalisms for computer-assisted improvisation. In *Proceedings of the Sound and Music Computing Conference*, 2010.

[21] Julian Moreira, Pierre Roy, and François Pachet. Virtualband : Interacting with stylistically consistent agents. In *Proceedings of the International Society for Music Information Retrieval*, pages 341–346, 2013.

[22] Jérôme Nika and Marc Chemillier. Improtek, integrating harmonic controls into improvisation in the filiation of OMax. In *Proceedings of the International Computer Music Conference*, pages 180–187, 2012.

[23] Jérôme Nika, José Echeveste, Marc Chemillier, and Jean-Louis Giavitto. Planning human-computer improvisation. In *Proceedings of the International Computer Music Conference*, pages 330–338, 2014.

[24] François Pachet. The Continuator : Musical interaction with style. In *Proceedings of the International Computer Music Conference*, pages 211–218, 2002.

[25] François Pachet and Pierre Roy. Markov constraints : steerable generation of Markov sequences. *Constraints*, 16(2) :148–172, March 2011.

[26] Charlie Parker and Jamey Aebersold. *Charlie Parker Omnibook*. Alfred Music Publishing, 1978.

[27] Stanisław A. Raczynski, Satoru Fukayama, and Emmanuel Vincent. Melody harmonisation with interpolated probabilistic models. *Journal of New Music Research*, 42(3) :223–235, 2013.

[28] Stanisław A. Raczynski, Emmanuel Vincent, and Shigeki Sagayama. Dynamic Bayesian networks for symbolic polyphonic pitch modeling. *IEEE Transactions on Audio, Speech and Language Processing*, 21(9) :1830–1840, 2013.

[29] Isaac Schankler, Jordan B.L. Smith, Alexandre R.J. François, and Elaine Chew. Emergent formal structures of factor oracle-driven musical improvisations. *Mathematics and Computation in Music*, 6726 :241–254, 2011.

[30] Erik M. Schmidt and Youngmoo E. Kim. Learning rhythm and melody features with deep belief networks. In *Proceedings of the International Society for Music Information Retrieval*, pages 21–26, 2013.

[31] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of the 1999 ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279–280, 1999.

[32] Greg Surges and Shlomo Dubnov. Feature selection and composition using pyoracle. In *Proceedings of the 2nd International Workshop on Musical Metacreation*, 2013.

[33] Richard S. Sutton and Andrew G. Barton. *Reinforcement Learning : An Introduction*. MIT Press, 1998.