Rapport du Projet de TOP : développer un logiciel de reconnaissance musicale

Charlotte Asselin-Boullé, Matta Karczewski et Lucie Neves

pour le 3 janvier 2017

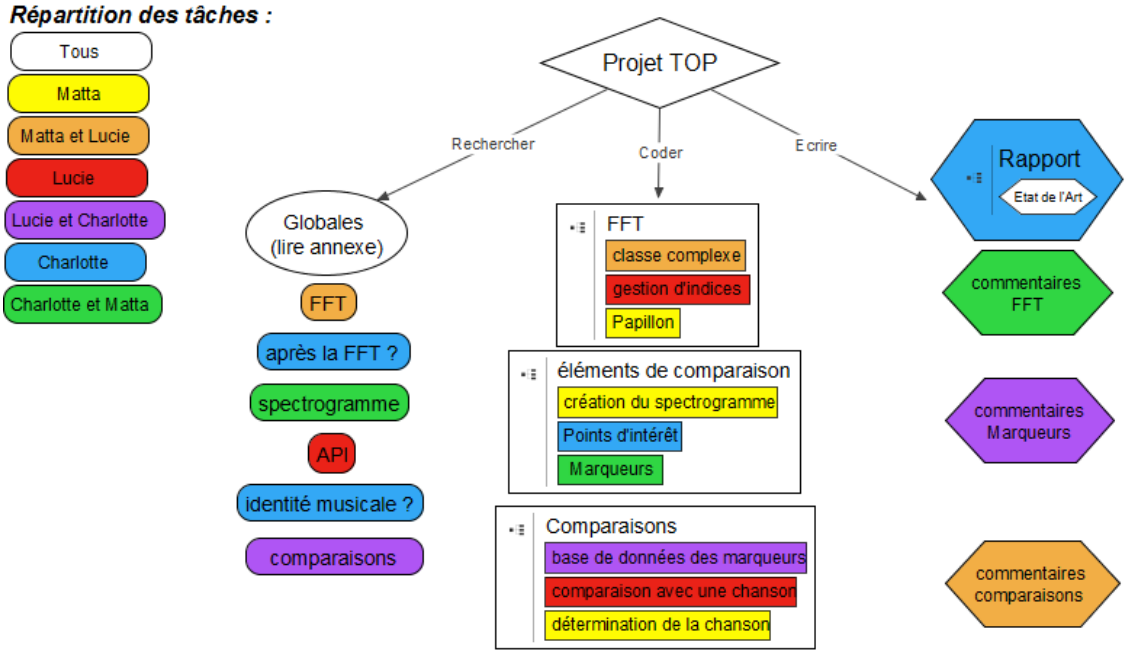
1 Introduction

Dans la cadre du projet de TOP de cette année, nous avons été amenés à concevoir un logiciel de reconnaissance d'échantillons musicaux en prenant comme modèle le logiciel Shazam.
Il nous fallait donc définir comme créer une empreinte acoustique, la comparer à la une base de données (avec plus ou moins de contraintes) et éventuellement proposer des services ou liens de services associés.
Nous sommes parvenu à créer un programme qui reconnait les échantillons fournis pour nos tests mais nous n'avons pas réussi à aller au delà.

2 Eléments de Gestions de Projets et répartition du travail

Pour notre projet nous avons utilisé divers outils introduits en cours de Gestion de Projet.

- Nous avons partagé et stocké nos codes sur la Forge, comme le préconisait les consignes.
- Nous échangions nos idées au travers d'un dossier partagé sur Google Drive.
- Le document soumis à votre lecture a été rédigé en LaTeX.
- Nous avons établie la carte conceptuelle ci-jointe que nous complétions si besoin au fur et à mesure pour se répartir le travail et déterminer ce qu'il restait à faire.



- Nous avons aussi établie une sorte de matrice RACI bien que nous essayions toujours de nous intéresser activement à l'avancé de chaque étapes (même celles ne nous concernant pas) en communiquant beaucoup (via les réseaux sociaux (FB et Skype), messages (SMS et mails), et appels).

Lot	Responsable	Recherches	Développeur(s)	Commentaires/Corrections
Global	Matta	Tous	Tous	Tous
Rapport	Charlotte	C et L	Tous	Tous
Fourier	Matta	M et L	M et L	M et C
Marqueurs	Charlotte	C et M	C	L et C
Correspondances	Lucie	C et L	L et M	M et L

Table 1: *Matrice de répartition des tâches*

Pour la répartition du travail, au début nous avions choisit de travailler essentiellement chacun de notre côté et de nous réunir régulièrement pour faire le point sur nos avancées et difficultés rencontrées. Finalement, ayant tous trois peu d'expériences dans la programmation, nous avons jugé qu'il serait plus simple et profitable pour nous trois d'avancer tous au même niveau et rester relativement sur les mêmes parties en parallèle. Ainsi nous nous tenions informés de ce qu'il se passait et pouvions nous aider.
La répartition du temps de travail a été la suivante :

- Nous avons passé beaucoup de temps sur la documentation. Notamment au début, n'ayant que très peu d'idées de comment (et quoi) faire. Tout trois avons passé un certain temps à chercher

des documents pertinents et croiser nos sources. Une fois le projet réellement lancé cette partie a plutôt été laissée au soin de Charlotte qui s'occupait en grande partie de la rédaction du rapport.

- Matta a passé énormément de temps sur la première partie du projet, assisté surtout de Lucie. Cette dite partie (étant essentiellement constituée de l'élaboration de la transformée de Fourier rapide) nous a demandé à tout trois beaucoup d'investissements, notamment pour la recherche et la compréhension de ce qu'il y avait à coder mais aussi pour le code en lui-même.

- La deuxième partie (constituée de la mise en évidence des marqueurs), beaucoup plus simple dans la programmation a, elle, demandé beaucoup de recherches mais aussi de réflexions pour l'optimiser. Elle a été attribuée à Charlotte, même si là aussi les trois ont participé puisque nous axions beaucoup nos décisions sur le partage d'idées.

- Enfin la troisième partie, dont le codage était là aussi assez ardu, été essentiellement à la charge de Lucie qui devait utiliser toutes les fonctions précédentes pour créer celle(s) qui cherche(nt) à faire correspondre les marqueurs de l'échantillon avec ceux de la base de données.

Le travail ainsi réparti nous a permis de passer environ autant de temps chacun sur le projet. Evaluer ce temps n'est pas forcément chose aisée puisque nous étions tantôt seul(e) à travailler dessus, tantôt à deux voire tous les trois, mais puisque nous devions le quantifier nous dirions que nous avons passé environ 60 heures chacun dessus (une petite dizaine d'heures avant les vacances, puis certains jours à 10h dessus d'autres à 3 ou 5h, et d'autres (notamment pendant les fêtes) pas du tout).

3 Etat de l'Art

Références	Sujets traités	Commentaires
An Industrial-Strength Audio Search Algorithm, Avery Li-Chun Wang – Annexe donnée avec le sujet	explication du fonctionnement global de Shazam	RAS
La Magie de Shazam : dans les entrailles de l'algorithme, Marie Georgescu de Hillerin *http://www.lesnumeriques.com/audio/magie-shazam-dans-entrailles-algorithme-a2375.html*	Résumé (en français) de ce que nous aurions à faire	pratique pour faire des mises au point et se remettre à jour sur ce qu'il reste à faire
How does Shazam work, Christophe *http://coding-geek.com/how-shazam-works/*	explications très détaillées du fonctionnement de Shazam avec beaucoup de points techniques	très complet (et long) mais bien fait (par parties) pour s'approprier nos parties (NB : en anglais)
Creating Shazam in Java, Roy van Rijn *http://royvanrijn.com/blog/2010/06/creating-shazam-in-java/*	rapide codage en Java	NB : anglais et Java
Understanding the FFT Algorithm, Jake Vanderplas *https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/*	Fourier	RAS
Fast Fourier Transform (FFT) *https://scala-lms.github.io/tutorials/fft.html*	Fourier	RAS
morceau de cours de M Tomczak	FFT	Base de notre codage. Utile pour la compréhension général de la fft
Apprendre Scala en Y minutes *https://learnxinyminutes.com/docs/fr-fr/scala/*	Commandes de base en Scala	RAS

4 Composition et conception de notre programme

En début de projet nous nous sommes réunit pour définir ce que nous souhaitions faire et comment. Ne sachant pas apprécier les difficultés de possibles fonctionnalités nous avons alors choisi qu'avant d'être ambitieux, l'idée était avant tout de rendre un travail cohérent et bien mené, même si " simple ". La question de l'interface graphique ne fut que très brièvement évoquée et rapidement mise de côté au vue de notre absence de connaissances en ce domaine. Nous préférions garder du temps pour réfléchir et améliorer la possibilité de reconnaissances, malgré le bruit par exemple.
Nous nous sommes imposés quelques contraintes comme :

- pouvoir reconnaitre la musique, quelque soit le segment en étude (pas forcément dès le début)
- pouvoir le reconnaitre même si il est endommagé (bruité) donc ne pas chercher à avoir tous les marqueurs identiques
- que la reconnaissance ne soit pas trop longue

Comme expliqué précédemment le projet fut découpé en trois parties.
La première partie fut la plus complexe du point de vue du code. Elle avait pour but de prendre en considération les signaux temporels fournis et de les transformer en spectral par le biais d'une transformée de Fourier.
La seconde cherchait à créer un spectrogramme puis en déterminer les points d'intérêt et alors créer un marqueur. Elle fut sujette à beaucoup de modifications puisque qu'il fallait définir ce qu'on qualifiait de " points d'intérêt " puis comme nous voulions définir nos marqueurs. Nous savions qu'il fallait travailler par bande de fréquences pour pallier le fait que notre audition soit logarithmique. Nous avons commencer par prendre le maximum de chaque bande puis vouloir prendre tous les maxima locaux de la bande quand ils dépassaient l'amplitude moyenne mais ceci faisait trop de points, nous sommes donc revenu à notre fonction ne prenant que les maxima. Pour les marqueurs il fallait aussi définir une fenetre adéquate, nous avons donc commencer par faire une fonction qui prenait un argument correspondant à la durée d'une zone cible. nous nous sommes alors rendu comtpe que dès qu'elle était supérieur à 0,3 seconde les marqueurs étaient suffisant, que quand ça dépassait 0,5 certes c'était plus précis mais beaucoup plus long nous avons donc statué sur une durée de fenêtre de 0,4 seconde.
Pour finir la troisième grande partie avait pour but de chercher les correspondances. L'idée était alors d'optimiser le plus possible cette recherche : il ne fallait pas qu'elle soit trop complexe car, même si notre base de données est très modeste, dans l'idée nous pourrions avoir beaucoup plus de musiques à parcourir. De plus il fallait définir le critère sur lequel nous allions déterminer que l'échantillon correspondait ou non à la musique pour qu'il ne nous retourne pas la mauvaise. Les textes parlant d'une comparaison " statistique " restaient très vague à ce sujet. Nous avons choisit de prendre le maximum mais tel qu'il y est une cohérence dans lesdécalage temporel des marqueurs correspondants. Par ailleurs nos premières versions du programme faisaient bien marcher le tout mais ils mettaient trop de temps (presque 4minutes pour notre avant dernière version) à établir la base de données avant de faire les correspondances il s'agissait alors de chercher des moyens d'optimiser.
Certains points se sont ajoutés, comme la gestion des musiques par l'API, mais aussi considérer la possibilité que l'échantillon soit en stéréo (et non mono-canal) ou bien sur une fréquence d'échantillonnage différente et donc la convertir pour éviter d'avoir des repliements de spectres... ou bien encore la gestion de la base de données qui nous a donné du fil à retordre lorsque nous pensions en avoir (enfin) fini or nous avons en toute fin decidé de copier la base de données dans un dossier de fichiers textes pour n'avoir à la calculer qu'une seule fois !

Le tableau suivant présente toutes les fonctions présentes dans le programme que nous avons rendu et leur complexité en fonction de l'entrée.

Nom de la fonction	entrée(s)	ce qu'elle fait	Remarques	Complexité
Complex	2 Doubles : la partie réelle et celle imaginaire	elle permet les calculs avec les complexes et de renvoyer un affichage standard	c'est la création de la classe des complexes	constante
butterfly	tableau de type Complex	renvoie la fft		$O(n * log_2(n))$ n taille du tableau
makeComplex	tableau de réels	retourne le même tableau en type Complex		$O(tailletableau)$
inver	tableau de taille $n = 2^p$	renvoie le tableau en changeant les indices	fonction basée sur le principe du tri fusion	$O(n * log(n))$
FFT	tableau de réels	renvoie le tableau de réels correspondant à la fft de celui en entrée	cette fonction combine celles vu précédemment	$n*log(n)$ n taille du tableau
StoM	tableau de tableaux (de taille n) d'entiers	renvoie le fichier en Mono	il n'agit que si le fichier entré est en stéréo	$O(n)$
harmonisation	tableau de tableaux (de taille n) d'entiers	renvoie le fichier échantillonné avec une fréquence de 11025Hz	on suppose que les fichiers sont échantillonnés avec une fréquence soit de 11025Hz soit 22050Hz soit 44100Hz	$O(n)$
empreintes	chemin du fichier et la taille des échantillons voulus	on retourne le spectrogramme ie le tableau des FFT des différents échantillons	on utilise l'API. On perd un tout petit peu d'informations si le fichier n'a pas une taille qui est multiple de la taille de l'échantillon	$O(m*n*log(n))$ avec m nombre de valeurs et n taille d'un échantillon
points_interet	tableau de réels (correspondant à une FFT à un temps donné)	retourne le tableau des fréquences où, pour ce temps, l'amplitude est remarquable	la sélection se fait par bande logarithmique de fréquence	O(taille du tableau)

marqueurs	tableau de tableaux de réels (correspondant au spectrogramme)	retourne les marqueurs (t ancrage, f ancrage, f , t-tancrage) après avoir appliquer la fonction points_interet sur chaque ligne du spectrogramme	les zones cibles ont une taille de 0,4 secondes	O(taille du tableau)
BasedeDonnees	chemin vers un fichier	stocke les marqueurs des chansons du dossier du chemin dans une variable globale		$O(nbchansons * m * n * log(n))$
match_sample	tableau de tableau de réels (les marqueurs) un indice (de chanson à tester) et le tableau de tableau de tableau contenant la base de donnée des marqueurs)	rechercher si il y a des correspondance entre la musique et l'échantillon et renvoie le maximum de correspondance suivant le décalage temporel cohérent		$O(lgechantillon * lgmusique)$
reconnaissance	un chemin et la base de donnée (tableau de tableaux de tableaux de réels)	renvoie une phrase disant si il a trouvé ou non une reconnaissance et si oui laquelle	cette focntion utilise toutes les autres	

Indexe des méthodes/fonctions et leur complexité

Pour utiliser notre programme il suffit de créer la base de données :
ecriturebase("/Users/matta/Desktop/test")

Ensuite pour tester un fichier il faut faire :
NomsChansons = Utils.listFiles("/Users/matta/Desktop/test")
println("testing little talks")
reconnaissance("/Users/matta/Desktop/musiques_top/WAV_Echantillons _connus_Mono_22050Hz/Little _Talks_Mono-22050Hz.wav","/Users/matta/Desktop/ecriture")

on a par exemple eu comme résultat :

```
TEST CONNU MONO 11KHz
testing laurie
La chanson que vous écoutez est Je_serai_Mono-11025Hz - copie.wav !
testing little talks
La chanson que vous écoutez est Little_Talks_Mono-11025Hz - copie.wav !
testing raiders march
La chanson que vous écoutez est The_Raiders_March_Mono-11025Hz - copie.wav !
TEST CONNU STEREO 44KHz
testing laurie
La chanson que vous écoutez est Je_serai_Mono-11025Hz - copie.wav !
testing little talks
La chanson que vous écoutez est Little_Talks_Mono-11025Hz - copie.wav !
testing raiders march
La chanson que vous écoutez est The_Raiders_March_Mono-11025Hz - copie.wav !
TEST INCONNU STEREO 44KHz
testing melissa
La chanson que vous écoutez est peut etre The_Path-11025Hz - copie.wav ?
testing ponies
Désolé, nous n'avons pas reconnu la chanson...
testing pirates
La chanson que vous écoutez est peut etre Little_Talks_Mono-11025Hz - copie.wav ?
```

Mystérieusement on a remarqué que ce n'est pas toujours le même fichier qui à le plus de correspondances puisque dans certain cas les fichiers en stéréo match avec ceux en mono et que tous les échantillonages testés match efficacement avec le fichier en 11025Hz correspondant.

Comme amélioration possible nous aurions pu faire en sorte que la recherche s'arrête si il trouve une très bonne correspondance pour que ça soit plus rapide dans l'absolue.

5 Difficultés rencontrées et enseignements

Les obstacles à notre progression ont été nombreux. Au-delà de la programmation nous avons eu à surmonter des problèmes techniques bien plus généraux que le codage en lui-même.

- Le premier obstacle à notre programmation fut de comprendre le fonctionnement d'un logiciel de reconnaissance musicale. Nous n'avions que de très maigres idées de ce qui pouvait se cacher derrière ces termes. Il s'agissait donc de se documenter. Nous avons passé énormément de temps à lire et relire encore et toujours des documents pour comprendre comment et pourquoi faire les choses. L'exemple le plus parlant étant la transformé de Fourier : il fallait déterminer enfin clairement ce qu'il y a derrière ce terme mystique puisque nous devions non plus l'appliquer " bêtement " mais le coder du début à la fin ! Nous avons ainsi appris à analyser des documents divers, les confronter, les comprendre en profondeur et jongler entre ce que chacun pouvait apporter pour refaire le puzzle des connaissances.

- Ces connaissances n'étaient pas forcément sur le code en lui-même mais aussi sur la manière de coder. Nous avons ainsi découvert plus en profondeur le codage en Scala avec lequel nous étions encore que peu familiers. Comme exemple nous pouvons citer les notions de classes et d'objets mais aussi la gestion des tableaux (affectations et distinction entre copier un tableau ou simplement pointer vers celui-ci) ou encore l'attention à apporter sur la nature des éléments que nous manipulions.

- Nous avons aussi compris pleinement l'importance d'être rigoureux dans nos codes puisqu'il fallait que nous trois (ainsi que le correcteur) puissions comprendre et même nous approprier et prendre en main ce que les autres avaient fait. Nous avons ainsi mit en place un dispositif de commentaires de nos codes que nous voulions très explicit pour ne rien laisser à l'interprétation, quitte à poser directement des questions entre nous dans ces dits commentaires et créer un dialogue interposé.

- Intervenait alors la gestion de notre équipe. Bien qu'une bonne entente soit nécessaire, elle n'est pas suffisante à l'établissement d'un projet à plusieurs. Il fallait alors s'organiser et pouvoir avancer malgré les contraintes de chacun, quelles qu'elles soient : éloignements des membres, emploi du temps (le gros du projet s'est tenu sur les vacances scolaires où nous avions tous des obligations, et l'envie de profiter des fêtes de fin d'année en famille !), habitudes de travail, impossibilité d'être joignable ou d'avoir une connexion internet,... Pour palier les difficultés engendrées nous avons donc établi les documents présentés plus haut et nous essayons de communiquer, énormément.

- Enfin l'un des freins le plus récurrent fut l'usage de la forge. Bien que nous pensions maîtriser git au travers des travaux et du partiel de gestion de projet, il s'est avéré que l'avancement de notre travail fut souvent soumis au bon vouloir des soucis que nous pouvions avoir dessus. Néanmoins avec rigueur et entraide nous avons fini par comprendre et apprivoiser cet outil.

6 Conclusion

Ce projet était avant tout un premier pas dans la programmation pour nous trois. Il fut donc fort intéressant du point de vue programme puisqu'il nous a permis de nous familiariser avec des éléments dont nous n'avions que peu (voire pas) l'habitude d'utiliser. Il nous a aussi aider à nous approprier le langage Scala de manière plus approfondie. Enfin et surtout il nous a fait découvrir ce que sont les projets dans des aspects moins formels. Ainsi nous avons beaucoup travaillé sur la gestion des projets, et le travail d'équipe qui en découle. Il nous a permis de nous rendre pleinement compte des bienfaits d'une bonne communication et d'une totale collaboration : il ne s'agissait pas seulement de bien faire la partie qui nous était assignée, mais de s'intéresser à tout le projet dans sa globalité pour comprendre ce que nous faisions et où nous allions. Bien que pas parfait et moins ambitieux que ce que nous aurions pu imaginer, nous sommes fiers de comment nous avons fonctionné et du fruit de notre travail. -