



Natural Language Processing for Fact-Checking and Claim Assessment

Intermediate report of project advancement

Othman EL HOUFFI
Dimitris KOTZINOS

MSc Research in Data Science & Machine Learning

December 12, 2021

Abstract

As false information and fake news are propagating though out the internet and social networks, the need of fact-checking operations becomes necessary in order to maintain a truthful digital environment where general information can be reliably exploited whether in politics, finance and other domains. The need of this online claim assessment comes from the fact that fake news and false information can have a big negative impact on politics, economy (2016 USA Elections) and public health (COVID-19).

A number of solutions have been proposed to deal with this problem and limit the spread of false information, both manual and automatic. Of course the manual approaches done on websites such as *PolitiFact.com*, *FactCheck.org* and *Snopes.com* don't construct a viable solution for the long term as the speed and scale of information propagation increase exponentially rendering this manual fact-checking operation where human fact-checkers can't scale up at the same rate limited and incapable of solving the problem.

Here, we present our contribution in this regard: an automated solution for fact-checking using Wikipedia's articles for claim verification. The algorithm uses NLP techniques in order to extract the so-called claim from the user input, then, using Wikipedia's API, it retrieves all the relevant articles and assesses with a degree of confidence if the claim is true, false or unable to decide due to lack of information showing evidence (sentences in articles) and probabilities for each resulted case.

Keywords: Natural Language Processing, Wikipedia, Information retrieval, Text processing, Natural Language Inferencing, Fact-Checking, Document retrieval, Sentence retrieval, Fake-news.

Contents

1	Introduction	3
1.1	Contexte du projet	3
1.2	Mise en scénario	3
1.3	Objectifs du projet	4
2	Présentation et spécification du projet	6
2.1	Fonctionnalités attendues	6
2.2	Conception globale du projet	6
2.3	Problématiques identifiées et solutions envisagées	7
2.4	Environnement de travail	8
3	Rendu final	9
3.1	Interface utilisateur finale	9
4	Conclusion	10
5	Perspectives	11

Chapter 1

Introduction

1.1 Contexte du projet

L'identification automatique de titres musicaux fait l'objet de nombreuses recherches, en particulier dans le cadre de l'indexation de larges bases de données et du monitoring de flux de broadcast.

Aujourd'hui les musiques font la base de toute plateforme diffusant un contenu multimédia, comme la radio, les chaînes de télévision, et bien-sûr les géantes plateformes que nous pouvons trouver sur internet telle que YouTube, Spotify, Facebook, Instagram, Netflix et autres. Parmi les problèmes communs que nous rencontrons sur toutes ces plateformes il y a le problème de la diffusion d'un contenu qui ne respecte pas les droits d'auteur, notamment le partage des musiques sans avoir ces droits.

Alors comment pouvons-nous détecter ce comportement illégal, ou plus précisément, existe-il un système capable d'identifier une musique par un segment d'audio ?

Cette question nous oblige à poser une série de questions ciblant le côté technique de ce système: comment pouvons-nous comparer un segment d'audio à une musique ? comment pouvons-nous réaliser cette comparaison sur une large base de données de musiques ? combien coûtera cette comparaison/recherche en termes de temps et de mémoire ?

Il existe plusieurs systèmes intelligents capable de réaliser cette tâche, et le plus connu s'agit de *Shazam*, ce système qui est aussi sous forme d'une application mobile qui peut être utilisée pour identifier une musique qui se joue dans votre entourage en utilisant le micro de votre smartphone en temps réel. Ce logiciel utilise le microphone du téléphone pour capturer un échantillon de musique jouée. Une empreinte acoustique est créée à partir de cet échantillon, elle est comparée à la base de données centrale de la société. Ce qui fait de *Shazam* un système très intelligent et largement utilisé, c'est l'implémentation de la technique du *fingerprinting*. Cette technique est basée sur l'extraction de pics spectraux qui sont associés par paires, ce qui permet de construire une constellation pour chaque signal.

Le 11 décembre 2017, Apple rachète *Shazam* sans donner d'indication sur le prix d'acquisition. Selon plusieurs sites spécialisés, le montant se situerait autour de 400 millions de dollars.

Le but de ce projet est de créer une application qui propose les mêmes fonctionnalités que *Shazam* tout en passant par toute les étapes nécessaires y compris l'échantillonnage du signal, le traitement de ce signal et la création voire aussi le stockage des empreintes acoustiques.

1.2 Mise en scénario

Supposons que vous entendiez une chanson dans un restaurant, dans un centre commercial ou dans une voiture, et que vous souhaitiez en savoir plus sur cette chanson. Par exemple, vous voulez connaître le titre de la chanson ou le nom de l'interprète ou de l'artiste. Les services de reconnaissance de musiques récents comme *Shazam* aident les utilisateurs dans de telles situations en identifiant l'enregistrement audio et en fournissant des informations appropriées sur le contenu. Un scénario typique est qu'un utilisateur, également appelé client, enregistre un court fragment audio de la chanson inconnue à l'aide d'un smartphone. Le fragment audio est ensuite converti en "empreintes audio", qui sont des caractéristiques audio compactes et descriptives. Ces empreintes sont transmises au

service d'identification, également appelé serveur. Le serveur héberge diverses ressources de données, notamment une base de données d'empreintes des morceaux de musiques connus, ainsi qu'une base de métadonnées qui contient des informations de contenu liées à ces enregistrements. Le serveur reçoit les empreintes digitales de requête envoyées par le client et les compare avec les empreintes digitales contenues dans la base de données. Cette étape est généralement réalisée par une consultation efficace de la base de données, soutenue par des structures d'index appropriées. En cas d'identification réussie, le serveur récupère les informations de contenu liées aux empreintes identifiées et renvoie les métadonnées souhaitées au client. La figure suivante présente un aperçu schématique du modèle client-serveur sous-jacent du service de fourniture de métadonnées décrit.

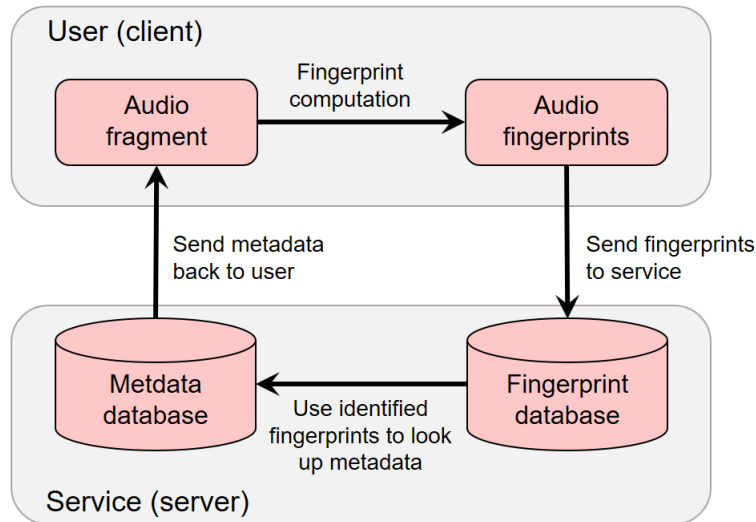


Figure 1.1: Client-Server Processing Pipeline

La tâche doit pouvoir être effectuée dans des milieux très perturbés sur des enregistrements de qualité médiocre, avec des échantillons de quelques secondes, en temps réel, avec peu de ressources computationnelles.

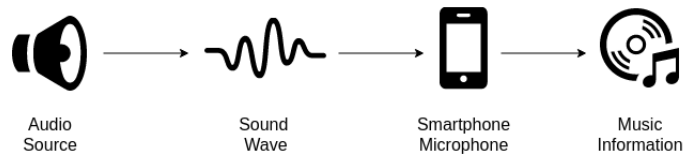
1.3 Objectifs du projet

La grande quantité de données disponibles et diffusées de manière continue sur tous les médias (radio, internet, télévision) pose le problème de l'exploitation efficace des contenus et du contrôle de sa diffusion. Dans le cadre du contrôle de flux multimédia, on cherche entre autres à identifier de manière robuste la donnée diffusée. Cette identification peut servir au contrôle des droits d'auteur, à la production de statistiques publicitaires, etc.

L'identification audio a pour but d'assigner son titre à une chanson diffusée. Dans le cadre de ce rapport, on s'intéresse à une identification par fingerprinting et par mise en correspondance d'une requête audio avec un élément d'une large base de données. L'identification par fingerprinting est basée sur l'extraction de caractéristiques qui décrivent de manière concise, unique et robuste les différents titres musicaux. Cette représentation est basée sur des propriétés acoustiques.

Pour cela nous avons besoin d'utiliser les différentes techniques de traitement de signal (échantillonnage, transformée de Fourier, spectrogramme, filtres...), ainsi que l'optimisation des requêtes pour une recherche rapide dans une large base de données.

L'application finale donnera une possibilité à l'utilisateur d'identifier une musique diffusée dans son entourage en utilisant le microphone de son appareil (ordinateur ou smartphone), ou bien à travers un segment audio sous forme mp3 qui servira comme un échantillon.



Chapter 2

Présentation et spécification du projet

2.1 Fonctionnalités attendues

Dans ce projet nous allons réaliser l'échantillonnage d'un signal acoustique à travers un microphone voire aussi à travers un fichier *mp3*, ensuite le traitement de ce signal afin d'extraire ses caractéristiques importantes, puis la création d'une empreinte associée à ce signal et finalement le stockage et la recherche des différentes empreintes dans une large base de données.

Une fois l'application aboutie, l'utilisateur aura la possibilité de :

1. Reconnaître une chanson à partir du microphone.
2. Reconnaître une chanson à partir d'un fichier *mp3*.
3. Traiter, Hacher, Stocker une ou plusieurs musiques dans la base de données.
4. Afficher les détails de la base de données.
5. Réinitialiser la base de données.

Dans notre programme, chaque chanson a une empreinte qui lui est associée. Quand on demande à notre programme de reconnaître un morceau, on décompose le son et le transforme en empreinte, puis on le compare à celles présentes dans sa base de données et on retourne une correspondance si elle existe.

Tout ces opérations (échantillonnage, traitement de signal, hachage, stockage et recherche) seront réalisées d'une manière scientifique voire technique où l'on étudiera des solutions existantes en regardant leurs avantages et leurs inconvénients, puis nous allons présenter une solution détaillée à chaque problème rencontré tout en certifiant sa validité avec des tests assez exhaustifs.

2.2 Conception globale du projet

L'identification audio se déroule en plusieurs phases :

1. Nous pré-calculons les empreintes digitales à partir d'une très grande base de données de morceaux de musique. Différents marqueurs peuvent être générés, mais ils sont souvent basés sur une analyse temps-fréquence du signal (*spectrogramme*).
2. Toutes ces empreintes digitales sont placées dans une base de données d'empreintes digitales qui est mise à jour chaque fois qu'une nouvelle chanson est ajoutée dans la base de données de chansons.
3. Lorsqu'un utilisateur utilise l'application, il enregistre d'abord la musique actuelle avec le microphone de l'ordinateur.

4. Pour un signal "requête", l'application applique l'algorithme d'empreinte digitale sur l'enregistrement de la même manière que pour les éléments de la base de données.
5. L'application vérifie si cette empreinte digitale correspond à l'une de ses empreintes digitales déjà présentes dans la base de données de chansons. Les algorithmes de mise en correspondance sont basés sur une recherche soit exacte, soit au plus proche voisin, soit statistique.
 - Si non, il informe l'utilisateur que la musique ne peut pas être identifiée.
 - Si oui, il recherche les métadonnées associées aux empreintes digitales (nom de la chanson, nom de l'artiste) et la restitue à l'utilisateur.

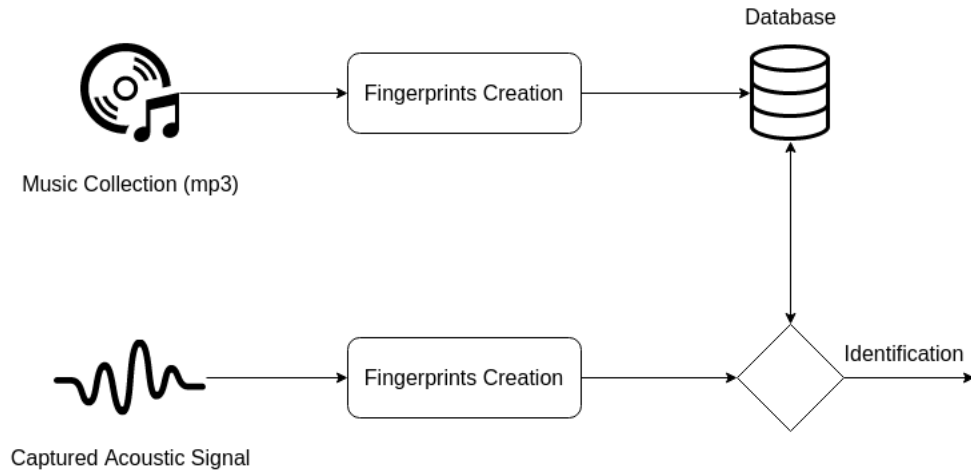


Figure 2.1: Processing Pipeline

2.3 Problématiques identifiées et solutions envisagées

Les systèmes de reconnaissance musicale du monde réel doivent être robustes et efficaces sur le plan informatique, ce qui entraîne un certain nombre de défis techniques à résoudre. En particulier, les empreintes audio utilisées dans ces systèmes doivent répondre à certaines exigences, notamment une spécificité, une robustesse, une compacité et une évolutivité élevées.

- **Spécificité** : Les empreintes audio doivent posséder une spécificité élevée, de sorte que même un fragment audio de quelques secondes seulement suffise à identifier de manière fiable l'enregistrement correspondant et à le distinguer de millions d'autres.
- **Robustesse** : Pour une identification fiable, les empreintes digitales doivent être résistantes au bruit de fond et aux distorsions du signal telles que la compression avec perte, le décalage de hauteur, la mise à l'échelle temporelle, l'égalisation ou la compression dynamique.
- **Compacité** : Les empreintes audio doivent être de petite taille afin de pouvoir être transmises sur des canaux à bande passante limitée et être facilement stockées et indexées du côté de la base de données.
- **Évolutivité** : Pour pouvoir s'adapter à des millions d'enregistrements, le calcul des empreintes audio doit être simple et efficace - une exigence qui s'impose également lorsque les empreintes sont calculées sur des appareils mobiles dont la puissance de traitement est limitée.

L'amélioration d'une certaine exigence implique souvent une perte de performance dans une autre, et il faut faire face à un compromis délicat entre des principes contradictoires. Par exemple, l'amélioration de la robustesse conduit généralement à une augmentation des identifications erronées (faux positifs), ce qui détériore la précision

du système d'identification. De même, même si elle est bénéfique pour des raisons de compacité et de calcul, une réduction excessive de la taille de l'empreinte digitale affecte négativement la capacité de discrimination. Inversement, les empreintes digitales d'une spécificité et d'une robustesse élevées peuvent ne pas être utilisables dans la pratique si leur calcul nécessite une puissance de traitement importante.

Les solutions que nous avons envisagé pour ces problématiques sont :

- Transformation du signal en *Spectrogramme*.
- Extraction des pics spectraux et construction d'une constellation.
- Formation des paires de pics spectraux et hachage combinatoire.
- Alignement des empreintes en utilisant un offset.
- Amélioration des requêtes et indexation des empreintes.

2.4 Environnement de travail

Nous allons utiliser le langage Python pour réaliser ce projet tout en profitant de plusieurs bibliothèques afin d'échantillonner, traiter le signal et hacher le résultat voulu.

Nous utiliserons un répertoire GitHub pour le partage. Concrètement, Git est un système de contrôle de version distribué, ce qui signifie que l'ensemble de la base du code et de l'historique est disponible sur l'ordinateur de chaque développeur, ce qui permet des branchements et une fusion faciles. Le contrôle de version nous aidera (développeurs) à suivre et à gérer les modifications apportées au code du projet. Au fur et à mesure le projet prend de l'ampleur, le contrôle de version devient essentiel.

Chapter 3

Rendu final

A travers ce projet nous avons réaliser une application qui permet de retrouver le titre d'une chanson et son auteur après seulement quelques secondes d'écoute par l'intermédiaire d'un microphone et aussi à travers un fichier mp3. Quand on demande à l'application de reconnaître un morceau, elle décompose le son et le transforme en empreintes. Puis le compare à ceux présentes dans sa base de données et trouve le résultat correspondant.

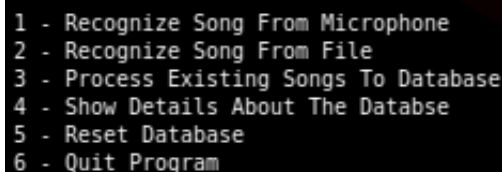
Pour cela nous avons procédé par le traitement de signal pour extraire des caractéristiques importantes, et nous créons ensuite une empreinte associée à ce signal afin de le stocker et finalement faire la recherche et la comparaison des différentes empreintes dans une large base de données.

3.1 Interface utilisateur finale

Par souci de temps, nous nous sommes limité à une version terminale, une interface interactive permet à l'utilisateur d'interagir avec un programme informatique, grâce à l'exécution du programme. Néanmoins, notre application ne demande pas de grands besoins du côté UI, une interface web ou mobile peut être réalisée dans un petit délai de temps en utilisant des frameworks moderne comme *ReactJS* ou *ReactNative*.

Au lancement du programme, six (6) options sont proposées à l'utilisateur qui sont les suivantes:

1. Reconnaître une chanson à partir du microphone.
2. Reconnaître une chanson à partir d'un fichier mp3.
3. Traiter, Hacher, Stocker une ou plusieurs musiques dans la base de données.
4. Afficher les détails de la base de données.
5. Réinitialiser la base de données.
6. Quitter le programme.



```
1 - Recognize Song From Microphone
2 - Recognize Song From File
3 - Process Existing Songs To Database
4 - Show Details About The Databse
5 - Reset Database
6 - Quit Program
```

Figure 3.1: Terminal Interactive Interface

Chapter 4

Conclusion

Ces dernières années, de nombreuses techniques différentes de création d’empreintes digitales et d’indexation ont été proposées et sont maintenant utilisées dans des produits commerciaux. Dans ce projet, nous avons examiné de plus près l’une de ces techniques, qui a été développée à l’origine pour le système d’identification audio *Shazam*. Nous avons discuté des idées principales qui sous-tendent ce système, mais il y a de nombreux paramètres qui doivent être ajustés afin de trouver un bon compromis entre les différentes exigences, notamment la robustesse, la spécificité, l’évolutivité et la compacité. Les aspects importants sont les suivants :

- les paramètres de la STFT (longueur de la fenêtre, taille du saut) qui déterminent les résolutions temporelle et spectrale,
- la stratégie de sélection et d’extraction des pics spectraux (avec ses paramètres de voisinage),
- la taille des zones cibles (utilisées pour définir les triplets), et
- des structures de données appropriées pour le hachage.

Bien que ce système est robuste à de nombreux types de distorsions du signal, l’approche de création d’empreinte discutée n’est pas conçue pour gérer les déformations temporelles. La correspondance des cartes de constellation ainsi que les différences d’horodatage (*timestamp*) dans les paires de pics sont toutes deux sensibles aux différences de tempo relatif entre la requête et le document de base de données. Par conséquent, il est nécessaire d’utiliser d’autres techniques pour être invariant aux modifications de l’échelle de temps.

Les empreintes digitales utilisant les pics spectraux sont conçues pour être très sensibles à une version particulière d’un morceau de musique. Par exemple, face à une multitude d’interprétations différentes d’une chanson par le même artiste, le système d’empreintes digitales est susceptible de choisir la bonne, même si elles sont pratiquement indiscernables par l’oreille humaine. En général, les systèmes d’identification audio sont conçus pour cibler l’identification d’enregistrements qui sont déjà présents dans la base de données. Par conséquent, ces techniques ne sont généralement pas généralisables aux enregistrements en direct ou aux performances qui ne font pas partie de la base de données.

Chapter 5

Perspectives

Comme décrit avant, il y a de nombreux paramètres qui doivent être ajustés afin de trouver un bon compromis entre les différentes exigences, notamment la robustesse, la spécificité, l'évolutivité et la compacité. Trouver des valeurs optimales à ces paramètres pourra augmenter largement les performances de notre application du point de vue de la robustesse aux distorsions, voire aussi du point de vue la mémoire utilisée et la vitesse de recherche d'une correspondance. Or, ce n'est pas une simple tâche, de plus les paramètres de notre application augmente, le processus de trouver des valeurs optimales à ces paramètres devient très compliqué.

Parmi les solutions que nous envisageons comme extension à notre application est l'utilisation d'un modèle de réseau neurones artificielles qui prendra en entrée les paramètres de notre application, et la sortie sera divisée sur les différentes exigences voulues tel que la robustesse, la mémoire, et le temps de recherche.

Ce réseau sera entraîné sur une large base d'apprentissage qui provienne de plusieurs tests déjà effectués d'une manière dynamique, par exemple nous allons exécuter la reconnaissance des morceaux sur une large collections de musiques tout en ajoutant du bruit et d'autres distorsions et aussi en variant le temps d'enregistrement du microphone, les résultats obtenus feront une très bonne base d'apprentissage pour notre réseau de neurones artificielles. Peut-être même on pourra ajuster les paramètres de notre application dynamiquement par rapport à chaque situation.

Bibliography

- [1] Avery L. Wang. An industrial-strength audio search algorithm. In *Proceedings of the 4th Symposium Conference on Music Information Retrieval*, 2003.
- [2] Peter Grosche, Meinard Müller, and Joan Serra: Audio Content-Based Music Retrieval. In *Meinard Müller and Masataka Goto and Markus Schedl (ed.): Multimodal Music Processing, Schloss Dagstuhl—Leibniz-Zentrum für Informatik*, 2012.
- [3] Audio Identification : https://www.audiolabs-erlangen.de/resources/MIR/FMP/C7/C7S1_AudioIdentification.html.
- [4] J. Haitsma, T. Kalker, and J. Oostveen, "Robust Audio Hashing for Content Identification". In *n International Workshop on Content-Based Multimedia Indexing*, 2001.
- [5] C.J. Burges, J. C. Patt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting". In *IEEE Transaction on Speech and Audio Proc*, 2003.
- [6] 6.050J/2.110J – Information, Entropy and Computation – Spring 2008
6.05<https://mtlsites.mit.edu/Courses/6.050/2008/notes/mp3.html>.
- [7] Seeing circles, sines, and signals <https://jackschaedler.github.io/circles-sines-signals/sound.html>.
- [8] Piotr Indyk, Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality.
- [9] Jerome Schalkwijk, A Fingerprint for Audio <https://medium.com/intrasonics/a-fingerprint-for-audio-3b337551a671>.
- [10] Jang et al. Pairwise Boosted Audio Fingerprint, 2009.
- [11] Short-Time Fourier Transform. In *Sensor Technologies for Civil Infrastructures*, 2014.
- [12] Nasser Kehtarnavaz. In *Digital Signal Processing System Design (Second Edition)*, 2008.