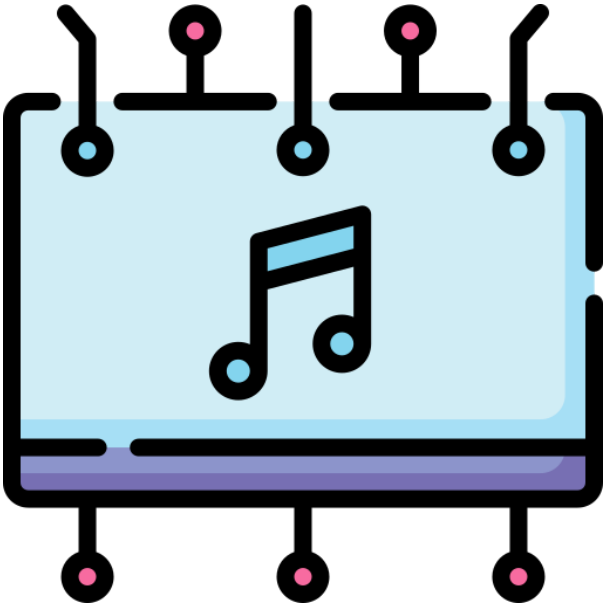
17 décembre 2021



RAPPORT CHALLENGE MEWO

[APPRENTISSAGE AUTOMATIQUE]

https://github.com/aperrier004/MEWO_BARBOSA_PERRIER

BARBOSA RÉMI - PERRIER ALBAN

BORDEAUX INP

ENSC/ENSEIRB 3A IA

SOMMAIRE

ANALYSE DU SUJET **2**
Contexte 2
Objectifs du projet 2
Données d'entrées 3
Le F1-score 3

DÉMARCHE DU PROJET **4**
Planning 4

EXPÉRIMENTATIONS ET ANALYSE DES RÉSULTATS **5**
Modèle Séquentiel 5
MultiOutputClassifier 5
Arbre de décision 5
RandomForestClassifier 6
Modèle avec données partitionnées 6
Les loss 7
Seuil par colonne 7

DISCUSSION **8**
Difficultés rencontrées 8
Bilan 9

RESSOURCES **10**

ANALYSE DU SUJET

Contexte

Ce challenge est proposé par l'entreprise Mewo, qui propose un service de gestion de catalogue destiné aux bibliothèques musicales de production. Sur leur plateforme, il est possible de parcourir différents catalogues musicaux et d'importer et stocker des données pour leur vitrine à destination de diffuseurs, régies publicitaires et maisons de disques. Leur solution permet la recommandation audio et le marquage automatique, basé sur un pipeline de traitement du signal et de réseaux de neurones profonds.

Leurs modèles génèrent des prédictions numériques (dans l'intervalle (0,1)) pour chaque étiquette sur laquelle ils ont été entraînés. Cependant, afin d'être véritablement utilisé, il faut associer, ou non, cette prédiction à un tag. Il est donc question de réaliser un processus de seuillage ou autres algorithmes plus élaborés pour permettre un étiquetage de qualité.

Objectifs du projet

L'objectif de ce challenge est ainsi de trouver la meilleure méthode de discrimination possible pour convertir les prédictions numériques en décisions efficaces. Mewo nous fournit ainsi un ensemble de données (décrit dans la partie suivante) où l'objectif est d'associer un ou plusieurs tags à chaque piste audio. Ce processus doit évidemment être automatique puisqu'il s'agit d'un problème de classification multilabels. Une piste est d'entraîner des réseaux de neurones qui permettront de générer des prédictions meilleures que celles du benchmark.

Notre classifier multilabels doit se baser sur toutes les informations disponibles, à savoir la vérité terrain ainsi que les prédictions générées par les réseaux de neurones de Mewo pour tous les tags et leurs catégories, pour produire un étiquetage des tags le plus juste possible.

Données d’entrées

Les données d’entrées fournies par l’entreprise Mewo comprennent quelques centaines de tags (blues-rock, electric-guitar, happy, etc.), regroupés en classes (Genres, Instruments ou Moods), partitionnés en catégories (genres-orchestre, instruments-cuivres, mood-dramatic, etc.).

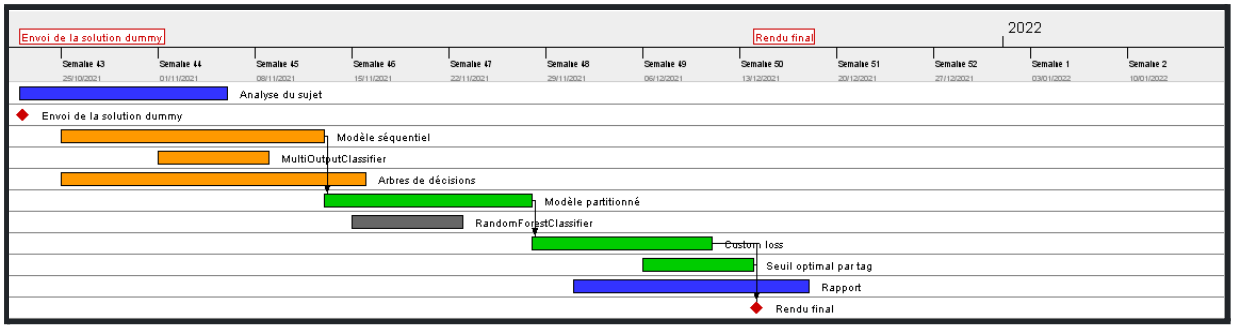
Le F1-score

Le score F1 est une mesure qui utilise la précision d'un modèle sur un ensemble de données. Il est utilisé pour évaluer les systèmes de classification binaires, qui classent les exemples en « positif » ou « négatif ». Le score F est un moyen de combiner la précision et le rappel du modèle, et il est défini comme la moyenne harmonique de la précision et de la sensibilité du modèle.

DÉMARCHE DU PROJET

Planning

Ci-dessous un aperçu global de notre organisation en tâches.



Nom	Date de début	Date de fin
Analyse du sujet	22/10/2021	05/11/2021
Envoi de la solution dum...	22/10/2021	22/10/2021
Modèle séquentiel	25/10/2021	12/11/2021
MultiOutputClassifier	01/11/2021	08/11/2021
Arbres de décisions	25/10/2021	15/11/2021
Modèle partitionné	13/11/2021	27/11/2021
RandomForestClassifier	15/11/2021	22/11/2021
Custom loss	28/11/2021	10/12/2021
Seuil optimal par tag	06/12/2021	13/12/2021
Rapport	01/12/2021	17/12/2021
Rendu final	14/12/2021	14/12/2021

: Tâche de recherche/rédaction

: Expérimentation concluante, mais avec des résultats trop faibles

: Expérimentation poursuivie et améliorée*

: Expérimentation n'ayant pas aboutie

Nous avons parallélisé nos expérimentations en travaillant sur des idées différentes, pour essayer d'en trouver une suffisamment efficace. Cela nous a permis de tester plus de choses rapidement. Notamment pour optimiser les ressources disponibles sur Google Colab, plateforme que nous avons utilisée pour exécuter nos modèles.

EXPÉRIMENTATIONS ET ANALYSE DES RÉSULTATS

Modèle Séquentiel

Une de nos premières expérimentations a été de créer un rapide modèle séquentiel, ce qui nous a permis de nous lancer sur le challenge et de notamment commencer à analyser les données que l'on manipulait. Ce modèle n'était pas voué à être gardé, il nous a plutôt servis de premier test pour nous familiariser avec le sujet.

MultiOutputClassifier

Nous voulions expérimenter d'ajuster un classificateur par cible. Il s'agit d'une stratégie simple pour étendre les classificateurs qui ne prennent pas en charge nativement la classification multicibles. Cependant, cela n'a pas été concluant, particulièrement avec le choix de l'estimateur. Au départ, nous avons essayé avec un *Support Vector Classification*, mais ce n'était pas adapté au nombre d'échantillons. Nous avons donc voulu utiliser un *SGDClassifier*, estimateur qui implémente des modèles linéaires régularisés avec apprentissage par descente de gradient stochastique (SGD), mais notre classification n'a pas été concluante, car il manquait un élément important pour que cela soit efficace sur notre problème de tagging : une forêt.

Arbre de décision

L'une des premières pistes fut d'utiliser un arbre de décision. Étant donné que les données d'entrée disposent d'informations sur les catégories, l'idée était d'entraîner un arbre de décision sur celles-ci afin de reconnaître la hiérarchie entre catégories et tags et d'apprendre les seuils de différenciation entre chaque label, pouvant éventuellement mener à une amélioration de la classification. Une fois entraîné, il suffit de faire une prédiction sur les données de test.

Notre premier arbre étant défini avec une profondeur maximum très grande (200), et les résultats n'étant pas au rendez-vous, nous suspections la présence d'overfitting. Cependant, même après avoir réduit la profondeur maximum à 30, les résultats restaient proches d'une stratégie appliquant un simple seuil à 0.5 (soit un F1-score de 33% sur les données de test). Étant donné la profondeur importante des arbres, nous ne les avons pas affichés avec graphviz.

RandomForestClassifier

Afin d'affiner la méthode précédente, on a envisagé une forêt d'arbres aléatoires constituée d'arbres s'occupant chacun de la classification d'une catégorie. Nous n'avons toutefois pas réussi à obtenir de résultats en raison d'une limitation mémoire.

Modèle avec données partitionnées

Une autre expérimentation que nous avons essayé d'améliorer petit à petit (puisqu'elle semblait augmenter nos résultats) était de réaliser un modèle avec plusieurs couches denses. Notre approche a également été de partitionner le dataset en fonction des tags et des catégories, nous donnant ainsi 6 couches d'input à traiter. Nous avons fait en sorte d'entraîner plusieurs modèles individuels, sur chaque couche d'inputs, pour qu'il soit entraîné sur une seule tâche. De plus, nous avons ajouté un Dropout sur les couches permettant de réduire l'overfitting lors de l'entraînement du modèle. Concernant l'entraînement du modèle, nous avons fait plusieurs expérimentations sur les hyperparamètres, notamment la taille du batch et le nombre d'epochs. Pour essayer de mieux comprendre leur influence, nous avons évalué le modèle selon les metrics de loss et val_loss, ce qui nous a permis de comprendre qu'augmenter indéfiniment le nombre d'epochs ne faisait que rallonger le temps d'exécution du modèle, avec une faible amélioration à la fin.

Les loss

Pour améliorer notre modèle final, nous avons expérimenté différentes custom loss. Tout d'abord de manière naïve, nous avions utilisé la *Binary crossentropy*, mais elle n'était pas adaptée, ce pourquoi nous avons recentré le problème avec une *F1-score metric*. Obtenant de meilleurs résultats, nous avons cherché si des améliorations étaient possibles, et en s'inspirant de ce que nous avons pu trouver nous avons implémenté une *f1_loss* et une *macro_double_soft_f1*. La meilleure fonction de perte serait en soit, la métrique elle-même. Cependant, le score macro F1 présente une limite importante, elle ne peut pas être différenciée. On ne peut alors pas l'utiliser comme fonction de perte. Ainsi, les deux fonctions que l'on utilise réalisent des modifications sur la F1-score pour qu'elles soient différentiables. Au lieu d'accepter des prédictions entières 0/1, elles acceptent plutôt les probabilités.

Seuil par colonne

Une autre piste était de définir un seuil optimal par tag (et donc par colonne). L'idée est de trouver pour chaque tag, le seuil qui maximise la précision et le rappel (et ainsi le F1-score) sur les données d'entraînements. On a implémenté cette approche de manière gloutonne (c'est-à-dire en testant tous les seuils possibles entre 0 et 0,6 avec un pas de 0,01) et peu optimisé. Sur les données de test, on obtient un score de 39,5%.

DISCUSSION

Difficultés rencontrées

La dimension exploratoire du projet a pu représenter une certaine difficulté en début de projet. En effet, le challenge est en place pour que le benchmark soit dépassé et que des élèves comme nous explorent différentes solutions afin d'obtenir des résultats performants. Nous avons donc essayé de bien prendre le temps d'analyser le sujet et les données que nous avions avant de commencer à sonder des solutions prometteuses, notamment parmi ce qui peut déjà exister aujourd'hui. Cependant, nous aurions probablement dû commencer et nous concentrer sur l'approche des seuils qui répond mieux au problème l'utilisation de modèle supplémentaire.

Un autre obstacle fut les temps d'exécution longs de nos solutions, ce qui nous faisait perdre un certain temps, particulièrement sur la prise de décision et la recherche de nos hyperparamètres. Cela était dû à notre méconnaissance de Google Colab en début et milieu de projet, puisque nous avons eu connaissance ensuite de l'environnement d'exécution GPU qui nous a permis d'accélérer l'exécution.

Dans une autre mesure, nous avions placé des espoirs certains dans les arbres de décisions et forêts aléatoires, deux solutions qui nous paraissaient très pertinentes au vu du problème. Seulement, nous n'avons pas su comment contourner le problème du nombre de noyaux à générer, qui dans le premier cas, c'est juste limité à ne pas afficher la hiérarchie, et dans le deuxième cas nous n'avons rien obtenu d'abouti.

Bilan

En conclusion, le principe général de notre première solution (*f1_loss*) est qu'elle divise les données en différentes parties correspondant aux différents types de données du dataset. Ces partitions sont traitées les unes avec les autres pour ensuite être concaténées. Cette étape nous permet d'utiliser toutes les données ensemble, pour que l'on puisse les ajouter aux données d'entrées. Notre seconde solution (*thresholds*) quant à elle recherche un seuil optimal pour chacune des tags. Ces deux approches sont nos expérimentations les plus abouties, nous donnant avec la première un score maximum de 0.4560 face au benchmark de 0.4561.
Nous retirons tout de même de ce projet une expérience *challengeante* et qui nous a permis de progresser en explorant des solutions diversifiées de ce qui peut nous être enseigné. Le caractère concret et appliqué du projet nous a également motivé à essayer de dépasser le seuil du benchmark, ajoutant ainsi un côté compétitif, mais nous n'avons malheureusement pas réussi à tenir nos objectifs et ambitions.
En somme, nos solutions sont encore à améliorer. Des éventuelles pistes sont à creuser concernant l'efficacité de notre solution de seuil.

RESSOURCES

- 1.10. Decision Trees. (2021). Retrieved 16 December 2021, from https://scikit-learn.org/stable/modules/tree.html
- Best loss function for F1-score metric. (2021). Retrieved 16 December 2021, from https://www.kaggle.com/rejpalcz/best-loss-function-for-f1-score-metric
- Challenge data. (2021). Retrieved 16 December 2021, from https://challengedata.ens.fr/participants/challenges/43/
- sklearn.ensemble.RandomForestClassifier. (2021). Retrieved 16 December 2021, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- The Unknown Benefits of using a Soft-F1 Loss in Classification Systems. (2021). Retrieved 16 December 2021, from https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d