Exploration interne de la structure conceptuelle des données

Maxime Haurel | L3 MIASHS parcours Sciences Cognitives | Mai - Juillet 2022
Sous la direction de : Mathieu D'Aquin

Loria
Laboratoire lorrain de recherche en informatique et ses applications

UNIVERSITÉ DE LORRAINE

Institut des sciences du Digital Management & Cognition
COMPOSANTE DE L'UNIVERSITÉ DE LORRAINE

Table des matières

Introduction

Le domaine de l'intelligence artificielle (IA) est en plein essor et n'a presque jamais connu d'hiver. Le nombre de recherches en IA n'a jamais été aussi conséquent et les applications de l'IA touchent tous les domaines, de la santé à l'agriculture en passant par l'éducation.

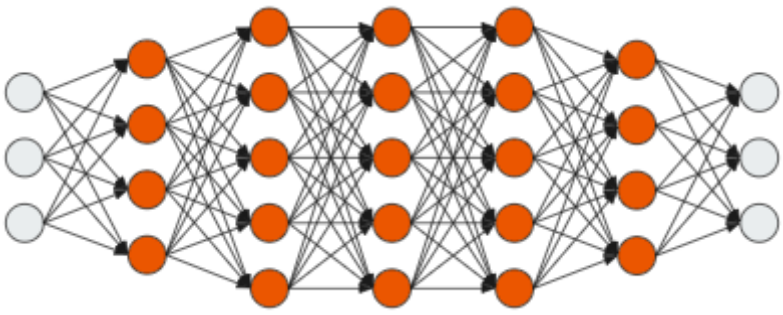
Alors que les systèmes experts utilisant des règles bien définies étaient les systèmes rois il y a 20 ans, aujourd'hui c'est les modèles statistiques qui sont les plus utilisés dans l'industrie et les plus connus sont les réseaux de neurones artificiels. Ces modèles fonctionnent très bien pour un grand nombre de tâches et c'est la principale tendance en IA connue sous le nom de Deep Learning.

Parmi les problèmes que le Deep Learning peut traiter, on peut identifier deux types de problèmes :

- La classification : le réseau essaye de ranger un individu dans une des classes que nous lui avons défini. La classification peut être binaire (ranger l'individu dans une classe parmi deux proposées) ou bien multiple (lorsque le nombre de classes est supérieur à deux).
- La régression : le modèle essaie de trouver la relation d'une variable par rapport à une ou plusieurs autres variables. Le but de la régression est d'estimer une valeur numérique à partir des entrées données au réseau.

https://projeduc.github.io/intro_apprentissage_automatique/regression.html#:~:text=La%20r%C3%A9gression%20sert%20%C3%A0%20trouver,ensemble%20de%20caract%C3%A9ristiques%20en%20entr%C3%A9e.

Bien que ces réseaux de neurones soient largement utilisés de partout, ils ont aussi un ensemble de désavantages dont un est **l'interprétabilité**. Les réseaux de neurones permettent d'abstraire les entrées reçues dans les couches cachées qui composent le réseau dont on peut en voir une représentation :



Ainsi, comme dans la psychologie behavioriste, on ne s'intéresse qu'aux entrées et aux sorties et les couches cachées du réseau (en orange ci-dessus) sont la boîte noire dont, à l'heure d'aujourd'hui, nous ne sommes pas en mesure de comprendre les rouages.

Ces réseaux peuvent par exemple être utilisés en production pour prédire le comportement d'un utilisateur. Mais bien que ce ces réseaux fonctionnent plutôt bien voire mieux que certains experts dans le domaine d'application, il serait bien venu d'être en mesure d'expliquer la décision faite par ce genre de réseau.

C'est donc presque dans cet intérêt que le stage s'inscrit. La vision de mon maître de stage est portée sur le comparaison des activations d'une couche cachée d'un réseau de neurones à une connaissance bien définie.
Dans cette optique, il est possible de penser plus loin en essayant par exemple de corriger le réseau sur des individus qu'il a mal classifié.

Présentation de l'entreprise

L'entité dans laquelle j'ai fait mon stage est le Laboratoire Lorrain de Recherche en Informatique et ses Applications, abrégé en LORIA et qui a été créé en 1997.
Le laboratoire se situe à Villers-lès-Nancy et représente un des plus grands laboratoires en recherche informatique sur le territoire français et un des plus grands laboratoires toutes thématiques confondues dans la région lorraine.

Au sein du laboratoire, on compte 400 personnes dont la plupart sont des ingénieurs et des chercheurs faisant partie de l'Université de Lorraine, de l'INRIA et du CNRS et les recherches sont divisées en 5 départements :

- Département 1 : Algorithmique, calcul, image et géométrie
- Département 2 : Méthodes formelles
- Département 3 : Réseaux, systèmes et services
- Département 4 : Traitement automatique des langues et des connaissances
- Département 5 : Systèmes complexes, intelligence artificielle et robotique

Quant à ma position dans le laboratoire, je me place en tant que stagiaire dans une des équipes les plus récentes, l'équipe K, dont le chef d'équipe est Mathieu d'Aquin.
L'équipe K est composée de :

- Mathieu D'Aquin
- Aurélie Bannay
- Jean Lieber
- Nicolas Jay
- Emmanuel Nauer
- Nicolas Lasolle (doctorant)

Cette équipe s'intéresse à l'intelligence artificielle symbolique et notamment aux systèmes basés sur les connaissances dont les principaux problèmes sont de savoir comment :

- Concevoir des moteurs d'inférence
- Construire des bases de connaissances utilisées par les moteurs d'inférence

L'équipe se centre donc sur la connaissance qui donne lieu à 3 domaines de recherche principaux :

- La science des données (ainsi que la découverte de connaissances)
- L'ingénierie des connaissances
- Le raisonnement (hypothétique et déductif)

Dans le cadre de leurs recherches, les membres utilisent majoritairement le langage Python qui est le langage par référence pour traiter des données.

Mon intégration à l'équipe s'est faite naturellement puisque les centres d'intérêts de l'équipe et les miens se rejoignent et mes compétences s'inscrivaient très bien dans l'environnement de l'équipe. J'ai pu fréquenter d'autres stagiaires de l'équipe qui travaillaient sur des sujets assez différents du mien.

Par ailleurs, je dirais que le LORIA a été très bénéfique dans ce sens car j'ai aussi pu rencontrer des stagiaires issus d'autres équipes, souvent de thématiques très éloignées, qui m'ont donné une ouverture d'esprit sur leurs domaines de compétences en ayant échangé avec eux.

Présentation du projet global

L'intuition scientifique de M. Mathieu d'Aquin a conduit à la création de ce stage. Cette intuition de pouvoir révéler des concepts au sein des réseaux de neurones artificiels à partir des valeurs d'activation des couches cachées est très inspirante et pourrait faire avancer de nombreux domaines de recherche.

PREREQUIS

Dans le but de pouvoir concrétiser cette intuition, certains prérequis étaient nécessaires, le principal étant de savoir programmer dans le langage Python. Heureusement, je connaissais déjà le langage mais les librairies que j'ai du utiliser m'étaient souvent inconnues.
De plus, malgré mon appétence pour le domaine de la recherche et les sciences dures de manière générale, lire des articles de recherche n'était pas aisé lorsque j'ai commencé ce stage.

ENJEUX DE L'INTERPRETABILITE

L'interprétabilité est une méthode permettant de fournir des informations concernant le raisonnement interne d'un algorithme de Machine Learning mais aussi sa représentation interne des données. Ces informations sont interprétables par des experts de la donnée mais ne seront pas comprises par des personnes externes au milieu.

Il existe une méthode proche appelée explicabilité qu'il ne faut pas confondre avec l'interprétabilité. L'explicabilité comprend l'interprétabilité dans son processus afin de rendre compte d'une manière très claire le fonctionnement interne d'un algorithme de Machine Learning.

Notre intention est d'interpréter les couches cachées d'un réseau de neurones en les comparant à des concepts.
Un concept est une représentation abstraite d'un objet ou d'un ensemble d'objets ayant des caractères communs[1].

Cela nous permettra selon le jeu de données, de comparer les activations d'une couche d'un réseau pour un individu à un concept représenté par la moyenne des activations de tous les individus appartenant à un concept et ce pour chaque couche.

Prenons un exemple de réseau utilisé dans le cadre du stage. Ce réseau prédit une classe de revenus d'un film en fonction de sa description textuelle, par exemple « Ratatouille is a 2007 American computer-animated comedy film produced by Pixar [...] later voted one of the 100 greatest motion pictures of the 21st century by a

[1] https://dictionnaire.lerobert.com/definition/concept

2016 poll of international critics conducted by the BBC. » sera classifié dans une des classes de revenus :

- « medium-low »
- « medium-high »
- « exceptional »

Une fois le réseau entraîné, on aimerait donc comparer les films à des concepts. Ce qui nous semble le plus pertinent est de comparer les activations au concept de *pays de provenance*.
Cela permettra de prendre pour chaque pays les activations des films associés et de visualiser les similarités entre pays ou entre individus. La procédure et l'implémentation seront détaillées dans les étapes qui vont suivre.

DESCRIPTION DU PROJET

On aimerait savoir quel est le cheminement de la décision d'un réseau de neurones puisque, comme je l'ai expliqué, il est aujourd'hui très difficile d'expliquer ne serait-ce qu'un peu ce qu'un réseau fait et impossible d'expliquer les concepts entrant en jeu dans la décision d'un réseau de neurones.

Les couches cachées d'un réseau se transfèrent entre elles des valeurs abstraites qui sont nommées des activations. Ensuite, des opérations mathématiques sont réalisées en fonction du type de la couche qui est traitée puis, une fois arrivé à la couche de sortie, ces valeurs sont lisibles par un humain et nous sommes enfin en mesure d'avoir une interprétation du résultat.

Pour le cas de mon stage, on aimerait savoir comment des concepts existants ou qui regroupent des éléments sont représentés au sein des réseaux de neurones.
On utilisera les valeurs d'activation afin de comparer les concepts ainsi que les individus entre eux et essayer de visualiser des similarités pour arriver à une conclusion.

Dans la continuité de la volonté de comparer les concepts et les individus et de pouvoir automatiser tout le processus d'exploration des couches d'un réseau, la conception d'un logiciel complètement automatisé qui permet la visualisation de concepts ainsi que l'exploration des activations pour un réseau quelconque ainsi que son jeu de données associé sera présenté.

TRAVAUX EXISTANTS

L'interprétabilité en IA est un domaine assez à la mode dans la communauté IA. Beaucoup d'articles sont publiés sur différentes techniques mais les articles qui

nous intéressent ici sont ceux traitant de concepts de haut niveau et des activations des couches dans les systèmes d'IA numérique/stochastique.

SCIENCE DES DONNEES

Les modèles statistiques de Machine Learning s'appuient sur des données et les réseaux de neurones encore plus. Il était donc nécessaire de récolter des données dans le but premier d'entraîner le réseau que je construirai et dans un second temps pour récolter des concepts que j'alignerai avec les valeurs d'activations du modèle.

Afin de récolter ces données, une multitude de jeux de données est disponible en ligne que l'on pourrait utiliser. On pourrait aussi utiliser une méthode de scraping pour récupérer des données publiques présentées sur des sites web.
Dans notre cas, on utilisera une autre méthode qui est d'utiliser une API et d'envoyer une requête pour accéder aux données qui nous intéressent. On utilisera ici l'API de DBpedia en écrivant une requête dans le langage SPARQL.

Les données étant récupérées sur DBpedia, elles ne sont pas prêtes à l'emploi. Elles ont besoin d'être traitées de manière réfléchie pour que le modèle les interprète correctement.
C'est notamment le cas des descriptions qui sont au format texte. On ne peut pas donner de texte à un réseau de neurones ou à tout autre modèle de Machine Learning puisque ces modèles sont avant tout des modèles statistiques donc qui fonctionnent avec des entrées numériques. Alors, afin de nettoyer le texte, j'ai appliqué des méthodes courantes de traitement automatique des langues comme :

- Transformer chaque lettre en minuscule
- Enlever la ponctuation, les liens, les entités numériques, ainsi que les mots très fréquents (stopwords) du genre *the, a, been, ….*

Précision : le jeu de données étant en anglais, les mots très fréquents sont évidemment les mots fréquents en langue anglaise.

Le texte a aussi besoin d'être tokenizé, c'est-à-dire d'être représenté sous forme de liste. Chaque description a donc la forme d'une liste qui contient comme élément chaque mot qui compose la description qui a déjà commencée à être traitée.

Le réseau n'accepte uniquement des entrées numériques. Or, nos descriptions sont encore au format texte. On utilise encore une fois une méthode provenant de keras afin de transcrire ce texte en séquences de nombres.

Il nous faut aussi ajouter du padding dans cette liste de séquences puisque toutes les descriptions ne font pas la même taille et le réseau n'acceptera pas plusieurs

entrées de tailles différentes. Ici, le padding est représenté par des zéros qui viennent s'ajouter au début de la liste jusqu'à ce que la liste fasse la bonne taille.

Ensuite, j'ai dû remanier les revenus pour créer les classes de revenus que j'ai évoqué plus tôt.
Les classes de revenus représentent la « cible » du réseau. Durant son entraînement, il va essayer en fonction de ses entrées, d'ajuster ses poids qui modulent les neurones au niveau des couches cachées afin d'améliorer sa précision à être fidèle à la « cible ».

Un modèle apprend sur un jeu de données et on évalue sa performance via une précision sur un jeu de validation. La différence est que le modèle ne connait pas les valeurs vraies (« cible ») des individus du jeu de validation et cela est logique puisque ce serait insensé de l'évaluer sur ce qu'il connait déjà.
On découpe alors le jeu de données traité en un jeu de données d'entraînement qui prendra 80% du jeu de données originel et donc les 20% restants du jeu de données originel vont pour le jeu de validation.

[... détail construction modèle ...]

Un des avantages du Deep Learning est de pouvoir créer un modèle tel qu'on le souhaite. Ainsi, il est possible d'ajouter les couches que l'on souhaite et qui correspondent évidemment au problème que l'on veut traiter.
Ces couches sont ajoutées à la suite comme si elles étaient empilées et se transfèrent les informations entre elles au travers de leurs neurones dont la taille est variable et c'est au constructeur du réseau de la spécifier.

Je rappelle que dans le cas actuel, nous voulons que le réseau traite des données correspondant à du texte et classifie selon des classes de revenus. Je crée donc un réseau de neurone dit *récurrent*. Ce genre d'architecture est utile pour le traitement de texte puisqu'au lieu d'apprendre de manière linéaire, ces réseaux comportent des cycles qui permettent la mémorisation des éléments passés. Ainsi le réseau pourra essayer de comprendre le sens global de la phrase en faisant le lien entre les mots qu'il rencontrera et les mots qu'il a rencontré.

[... expliquer construction pas à pas des couches ? ...]
La construction du réseau est donc très dépendante du problème auquel nous faisons face. Je vais expliquer dans le cas pour lequel je prends l'exemple depuis le début, c'est-à-dire la prédiction des revenus à partir de descriptions texte de films.

[... entraînement du réseau ...]

EXPLORATION CONCEPTUELLE

Maintenant que nous avons un réseau entraîné dont les performances sont assez raisonnables pour continuer, nous pouvons enfin voir ce qu'il se passe à l'intérieur du réseau.

Visualisation

Pour trouver des concepts dormant dans les couches du réseau et plus précisément au sein des valeurs d'activations des neurones de chaque couche, on voudrait visualiser la représentation interne d'un concept quelconque pour voir si des patterns seraient identifiables, ce qui nous rapprocherait de l'hypothèse de départ.

On a alors besoin d'un outil de visualisation qui pourrait nous montrer que des patterns représentant un concept précis existent au sein des neurones des couches du réseau. Cet outil de visualisation a besoin d'être clair et de communiquer directement en le regardant l'existence du concept.
Les valeurs d'activations étant déjà cloisonnées dans l'intervalle [-1 ; 1], il n'est pas nécessaire de les standardiser pour les visualiser.
Dans les activations du réseau un concept pourrait être représenté par des valeurs d'activations similaires sur un ou plusieurs neurone(s) pour chaque individu appartenant au concept.

La première solution conceptuelle trouvée était de représenter la moyenne par neurone des activations des individus appartenant à un concept *x* dans un nuage de points. Ainsi on pourrait voir si certains groupes de neurones sont caractéristiques à des concepts.

[... scatter plot of country in bycountry_ds ...]

[... scatter plot of sample in country in bycountry_ds ...]

Une deuxième solution est de représenter ces activations sous la forme de heatmaps. Ces heatmaps nous permettent de représenter les valeurs grâce à des couleurs. De cette manière, il sera bien plus facile de voir des patterns que dans les nuages de points. **[différence a la moyenne]**
Les heatmaps sont hautement personnalisables et cela nous a permis de modifier les couleurs afin d'avoir une bonne échelle pour que les couleurs parlent d'elles-mêmes. Ainsi, plus les activations seront proches de -1, plus elles seront couleur cyan et plus elles seront proches de 1, plus elles seront couleur jaune vif.

[... heatmaps of country in bycountry_ds ...]

[... heatmap of sample in country in bycountry_ds ...]

On voit que les heatmaps représentent mieux les nuages de points qui sont plus difficiles à interpréter. Je parlerai plus tard dans le rapport de l'implémentation de la visualisation sous forme de heatmaps.

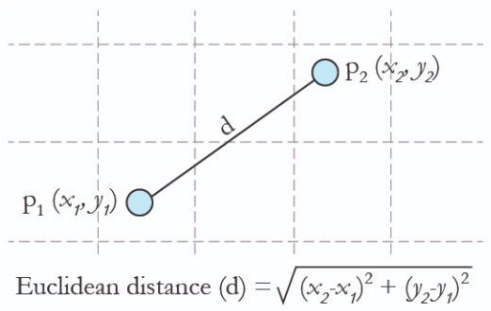
Mesure de la similarité

On peut dorénavant comparer à l'œil nu les activations entre concepts et individus. Ce qui fait sens ici est de voir si un individu appartenant à un concept– disons « France » - est réellement proche de cette catégorie en terme d'activations. On pourrait

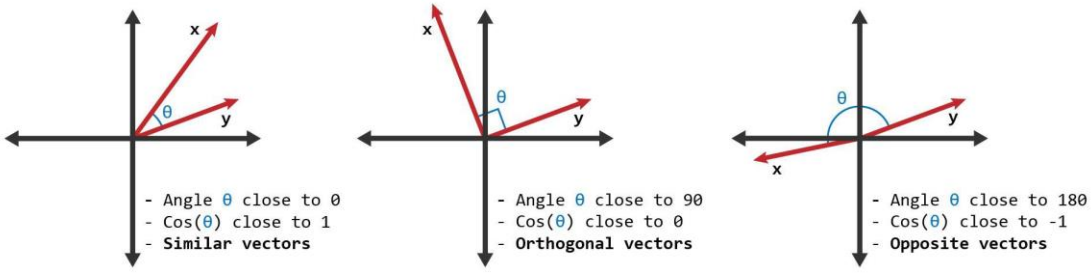
simplement regarder les heatmaps et valider ou non les similitudes. Néanmoins, on aimerait une mesure plus fiable et c'est pour cela que j'introduis l'on aimerait inclure une métrique de similarité dans la comparaison.

Etant donné que les activations sont représentables sous forme de vecteurs, les deux mesures de similarités auxquelles j'ai pensé sont :

- La distance euclidienne



- La similarité cosinus



Ces mesures de similarité nous donneront des valeurs plus concrètes que le simple fait de regarder pour des schémas (ou patterns) dans la visualisation des heatmaps.
On aura alors une représentation fiable par couche d'à quel point deux vecteurs sont similaires donc à quel point un individu ressemble à un concept ou à quel point un concept est similaire à un autre.

Présentation du travail réalisé

Je vais détailler ici la réalisation puis l'implémentation de chaque solution conceptuelle trouvée aux problèmes dont j'ai fait face dans l'optique de la réalisation de la problématique de ce stage.

SCIENCE DES DONNEES

Récolte des données

Les données abondent sur Internet, il est donc plutôt aisé de récupérer des données pour satisfaire une tâche. Il existe aussi plusieurs méthodes mais celles que nous avons choisi est d'appeler l'API de DBPedia avec une requête SPARQL, langage dont j'avais eu vent pendant un cours dispensé par mon tuteur de stage.
Voici donc la requête :

```
select distinct ?film ?income ?cat ?desc where {
?film a <http://dbpedia.org/ontology/Film> .
?film <http://dbpedia.org/ontology/gross> ?income .
?film <http://dbpedia.org/ontology/abstract> ?desc .
  {
    ?film <http://purl.org/dc/terms/subject> ?cat .
    } UNION {
      ?film <http://purl.org/dc/terms/subject> ?scat .
      ?scat <http://www.w3.org/2004/02/skos/core#broader> ?cat.
  }
  filter (lang(?desc) = "en")
  filter (lang(?film) = "en")
} group by ?film ?cat ?desc LIMIT 10000 OFFSET 100000
```

Cette requête récupère, pour l'ontologie Film sur DBPedia, son revenu, sa description textuelle en anglais, la ou les catégorie(s) cinématographique(s) à la quelle il appartient ainsi que son titre en anglais aussi.

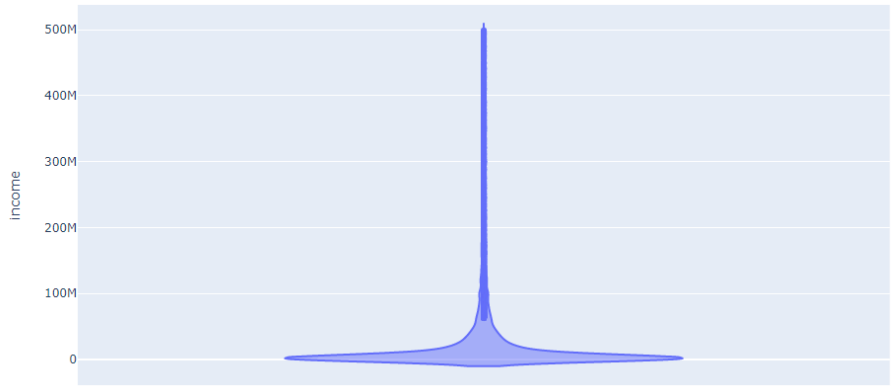
Par défaut, l'API de DBPedia retourne un nombre d'élément maximum égal à 10000 et ce n'est qu'après plusieurs requêtes et des recherches Internet que je l'ai découvert. C'est donc pour cela que la dernière ligne de la requête présentée ci-dessus intervient. Elle demande a DBPedia de renvoyer un nombre d'élément de taille 10000 et à partir du 10000ème élément *(offset)* et j'ai répété cette requête jusqu'à avoir 50000 éléments donc 5 fichiers contenant les données que je vais remanier par la suite pour avoir un seul et même fichier soit un grand tableau de données où chaque ligne serait un film.

Ce besoin de séparer les revenus en 3 classes distinctes relève du fait que nous ne pouvons pas prédire une valeur bien définie pour chaque film, c'est-à-dire faire une régression puisque les valeurs sont trop disparates.

Traitement des données

Pour l'exploration et le traitement des données, j'ai choisi d'utiliser le langage Python ainsi que les outils Jupyter Notebook et Google Colab.
Il paraît aujourd'hui évident d'utiliser Python pour ce genre de tâche puisque le langage est très en vogue et de nombreuses librairies pour ce genre de tâche sont disponibles.
Jupyter Notebook et Google Colab permettent de créer des *notebooks*, ce sont des fichiers dans lesquels il est plus simple de travailler puisque l'on exécute le code dans des cellules au lieu d'exécuter tout un code complet. Aussi, ces notebooks peuvent inclure des images, du texte brut et d'autres médias donc il est très simple de les mettre en page pour présenter des résultats ou des problèmes que j'ai rencontré.

Après avoir exploré les données rapidement, on se rend compte que les revenus des films sont assez disparates.



Sur cette représentation graphique en violon, on se rend compte qu'un très grand nombre de films ont leur revenus répartis dans le grand bassin situé approximativement entre 0 et 10000.
Cette représentation nous aide grandement dans le traitement des données puisque nous allons pouvoir créer nos classes de revenus.

On aimerait intuitivement regrouper les individus réparti dans le grand bassin dans le bas de la figure mais il y a tellement d'individus dans cette zone que l'on va préférer couper ce bassin en deux en prenant la médiane comme valeur de définition sectionnant les deux parties du bassin. La partie au-dessus du bassin, quand à elle, représentera une classe de revenus à elle seule.
Si on reprend, on a donc les classes :

- « medium-low » : [0 ; valeur médiane[
- « medium-high » : [médiane ; 10000]
- « exceptional » :]10000 ; +∞[

Le choix des classes est très important puisqu'il faut qu'elles ait du sens pour que le réseau aie des bonnes performances de classification dans les prochaines étapes.
Dans mon cas, les classes on été choisies avec réflexion donc on s'attend à avoir de performances plutôt raisonnables.

Une prochaine étape dans le traitement des données était d'enlever les *outliers* ou valeurs aberrantes.
En calculant le *z-score* qui est le nombre d'écart-type par rapport à la moyenne sur la distribution des revenus, j'ai choisi de définir comme *outliers* les individus pour lesquels le z-score pour leur revenu était supérieur à 2 et je les ai simplement enlevé de la distribution.

Ensuite, j'ai utilisé la librairie NLTK qui permet de faire une partie importante du travail de traitement de texte décris précédemment (mettre en minuscule, enlever les stopwords, tokenizer). La librairie keras vient aussi s'ajouter afin de transformer les phrases tokenizés en séquences numériques et d'ajouter le padding à ces séquences.

Après, j'ai dû remanier les revenus pour créer les classes de revenus que j'ai évoqué plus tôt. Puisque le modèle ne fonctionne qu'avec des valeurs numériques, les classes de revenus *(medium-low, medium-high et exceptional)* ont été encodées de la manière suivante :

- On crée une colonne dans le jeu de données pour chaque classe de revenus
- Pour chaque film et chaque colonne des classes de revenus on place un 0 si le film ne fait pas partie de cette classe, ou un 1 si justement le film fait partie de cette classe de revenus

On a alors un tableau présenté tel que :

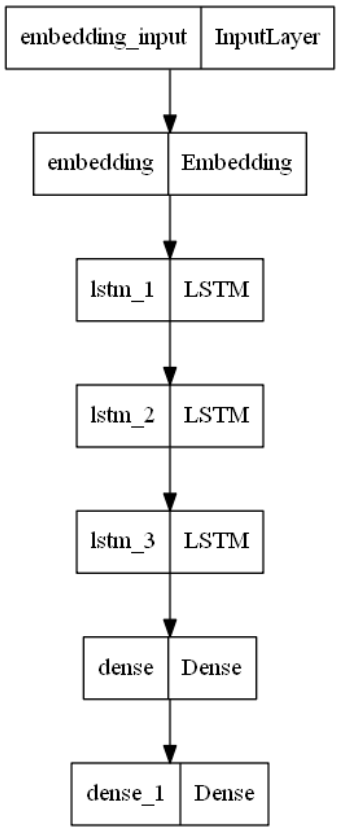
	exceptional	medium-high	medium-low
0	0	1	0
1	0	0	1
2	0	1	0
3	1	0	0
4	0	1	0

Ce tableau représente la « cible » du réseau. Durant son entraînement, il va essayer en fonction de ses entrées, d'ajuster ses poids qui module les neurones au niveau des couches cachées afin d'améliorer sa précision à être fidèle à la « cible ».

J'utilise la librairie de machine learning scikit-learn afin de partitionner le jeu de données en un jeu d'entraînement et un jeu de validation. Comme expliqué, il nous faut un jeu d'entraînement assez important pour être représentatif des données tout en ayant un jeu de test assez important pour avoir une valeur de précision fiable pour attester des performances de notre modèle.
On découpe alors le jeu de données traité pour avoir 80% en tant que jeu d'entraînement et les 20% restants vont pour le jeu de validation.

Construction & Entraînement du Modèle

Grâce à la librairie Keras venant de Tensorflow, j'ai pu construire un modèle représenté tel que :



Un réseau est composé de plusieurs couches et j'ai choisi d'utiliser les couches dont les noms sont marqués sur l'illustration ci-contre.
Je vais donc détailler l'utilité de chaque couche afin de justifier le choix de cette architecture.

La couche d'Embedding transforme chaque index de mots en vecteur d'embedding, c'est-à-dire en une représentation vectorielle des mots.

Puisque l'on veut que notre réseau traite des descriptions textuelles, on souhaite construire un réseau de neurones récurrent. Cela signifie que le réseau sera capable d'apprendre en tenant compte de ce qu'il a déjà appris en ayant des boucles au sein même des couches spécialisées en récurrence dont la couche LSTM, anagramme de *Long-Short Term Memory*, soit une couche présentant une mémoire interne.

Pour finir, les couches Dense sont des couches très basiques combinant les valeurs d'entrée ainsi que les poids et donnant en sortie une ou plusieurs valeurs (suivant le nombre de neurones) qui dépendent directement de la fonction d'activation associée.

[parler des fonctions d'activations ??]

Notre modèle est maintenant construit selon l'architecture présentée ci-dessus et il nous faut donc l'entraîner. Malheureusement il n'existe pas de meilleure méthode précise à suivre pour entraîner un réseau de neurone, il existe seulement des bonnes pratiques mais chaque jeu de données peut comporter des biais différents et il faut donc tester avec des paramètres différents à chaque entraînement pour essayer d'obtenir la meilleure précision possible.

Les paramètres sur lesquels je dois jouer sont le taux d'apprentissage, la fonction de coût, la taille du lot d'individu avant que le modèle se mette à jour et le nombre d'itérations d'entraînement du modèle.

Voici les paramètres qui m'ont donné la meilleure précision :

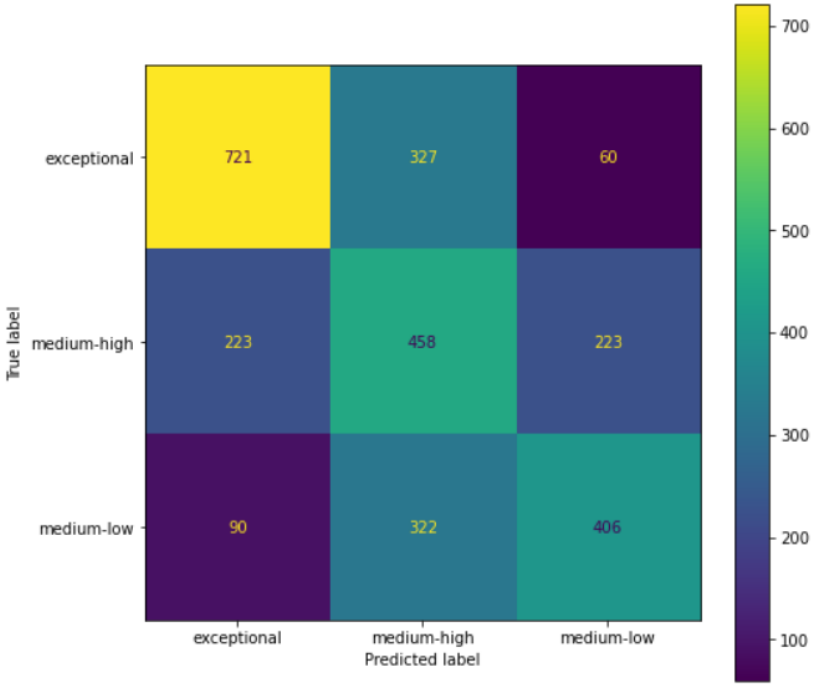
- Taux d'apprentissage : 0.0001
- Fonction de coût : « categorical_crossentropy »
- Taille du lot : 64

- Itérations d'entraînement : 30

A la fin de l'entraînement, on peut accéder à la précision que le modèle a eu sur le jeu de validation, ici nous avons une précision 56%.

La précision de notre modèle ne semble pas très satisfaisante a première vue. Néanmoins, il faut rappeler que la classification n'est pas binaire mais multiple. Le modèle arrive quand même plutôt bien à prédire les classes de revenus malgré qu'il existe 3 classes et que les classes *medium-low* et *medium-high* sont très ressemblantes.

Pour montrer visuellement ce que le réseau a prédit, il est judicieux de calculer et d'afficher la matrice de confusion.



Ce qui nous intéresse vraiment sur une matrice de confusion est la diagonale partant d'en haut à gauche puisque plus jaune elle sera, plus le modèle aura été précis dans sa classification.

A partir de ceci, on peut assumer que le réseau est assez performant pour avoir en ses activations, une représentation d'un concept qui serait par exemple, le genre cinématographique du film ou le pays de provenance du film.

EXPLORATION DES ACTIVATIONS DU MODELE

La direction de la recherche de la présence de concepts au sein même du réseau de neurones a été orientée par l'intuition que les activations d'une couche cachée d'un réseau pouvaient être porteuses d'un ou de plusieurs concept(s).

Maintenant que nous avons un modèle entraîné à notre disposition, on va pouvoir analyser ses activations. Mais avant cela, il nous faut les récupérer.

Récupération des activations

Il n'existe pas de fonction préconstruite pour afficher les activations d'une couche d'un modèle au sein des librairies Tensorflow et Keras. Et pour cause, il suffit de reconstruire le réseau jusqu'à la couche que l'on veut explorer et on re-exécute des prédictions avec les entrées originales pour obtenir les activations.

De cette manière il semble possible d'automatiser très facilement ce processus d'obtention des activations pour chaque couche et ce peu importe le modèle tant que l'on a accès aux données qui ont servies à son entraînement.

Traitement des activations

On va vouloir comparer des activations entre elles par la suite. Il nous faut alors les standardiser pour avoir une des données distribuée de la même manière.

$$X_{standard} = \frac{X - \mu}{\sigma} ,$$

Lorsque l'on voudra visualiser les activations pour un concept en particulier, il nous faudra prendre la moyenne de chaque neurone des activations associées à ce concept et on répète ce processus pour chaque couche.

Pour ce qui est des individus, il nous suffira de récupérer les activations puis de les afficher sous la forme que l'on désirera.

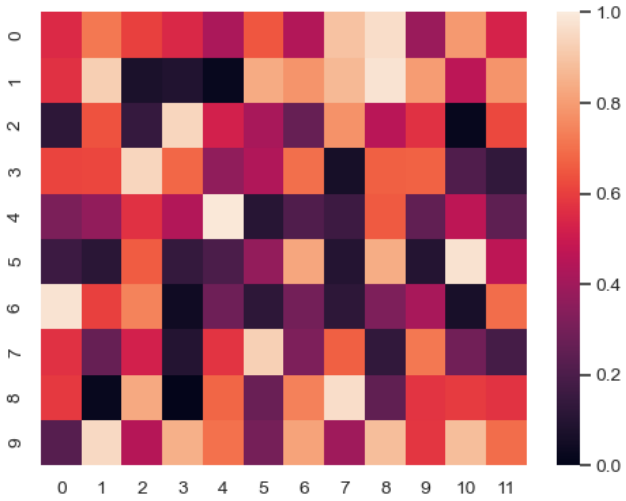
C'est au niveau de la couche *Embedding* que les processus vont être différents. En effet, cette couche nous retourne un vecteur de dimension 2 pour chaque individu. Ainsi, dans notre cas, pour chaque individu, il nous est retourné un tableau de taille 150x64. On ne peut pas standardiser les activations simplement, il faut alors remettre ce tableau à deux dimensions « à plat » et ce, pour chaque individu. Alors, pour chaque individu, les activations sont un tableau à une dimension comprenant 150*64 éléments soit 9600 éléments.
Le tableau regroupant tous les individus et leur 9600 activations peut maintenant être standardisé normalement.

A chaque fois que nous voudrons accéder aux activations d'un individu, il nous faudra remettre les activations à plat vers un tableau originel de la forme 150x64.
Ensuite, pour les 64 neurones, on prendra la moyenne des activations de chaque neurone.

Le principe est le même pour les activations associées à un concept. Il faudra simplement faire le traitement par individu que je viens d'expliquer mais pour chaque individu inclus dans le concept.

Heatmaps

Une heatmap est une technique de visualisation qui permet de visualiser plus efficacement les différences au sein d'une collection via l'utilisation de couleurs représentant l'intensité d'une valeur pour chaque valeur dans la heatmap.



Voici une heatmap basique qui représente des données quelconques.

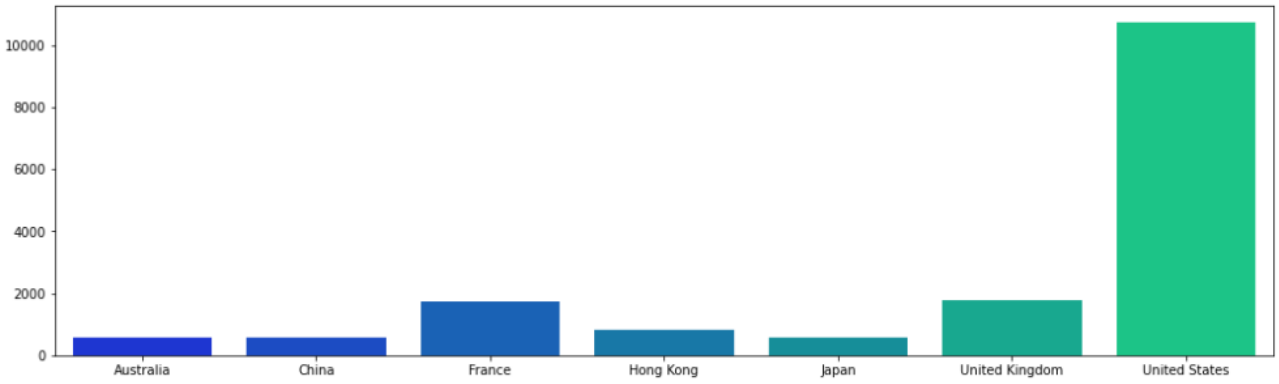
On voit grâce à la légende présente sur le côté droit que plus une case est blanche, plus la valeur y étant représentée est proche de 1.0.

Ce qui fonctionne plutôt bien dans notre cas puisque l'on espère visualiser des différences entre les neurones d'une catégorie/individu mais aussi de voir à l'œil nu les similarités entre un individu et la catégorie qui lui est associée par exemple.

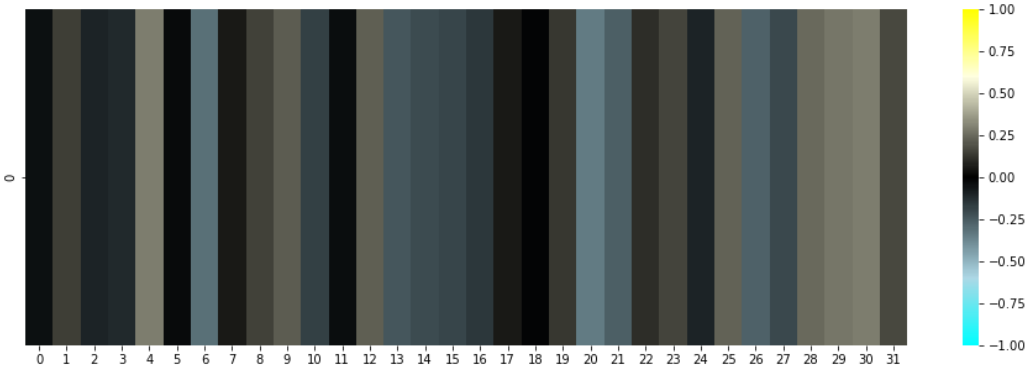
Les heatmaps sont générées par le biais des librairies matplotlib et seaborn qui fonctionnent ensemble et offrent une multitude d'outils de visualisation très pratiques pour la science des données.

Etant donnée la multitude de genres cinématographiques différents, j'ai choisi de ne prendre que les plus populaires et donc d'exclure les genres étant sous un certain seuil, seuil qui correspond au nombre de films inclus dans le concept auquel on s'intéresse, ici le genre cinématographique.

Rapidement, en essayant plusieurs seuils, 500 semble être une bonne valeur de référence puisque l'on se retrouve avec assez de concepts différents pour faire des comparaisons intéressantes.

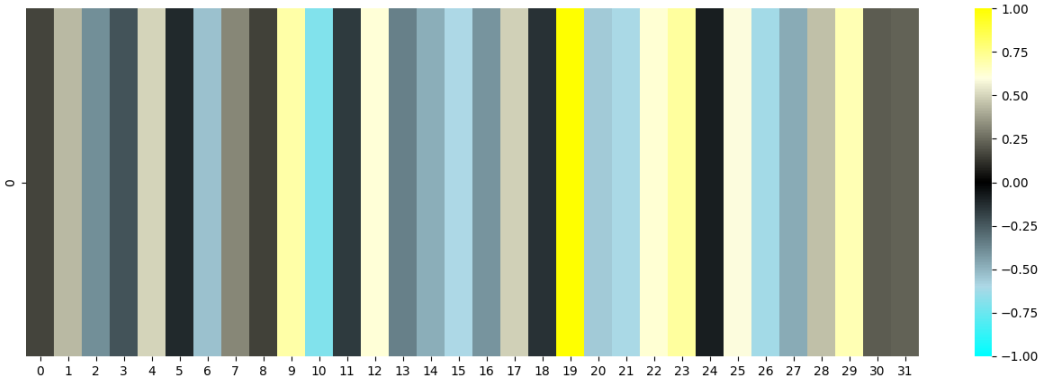


On peut donc visualiser la heatmap des activations pour un concept et pour une couche spécifique du réseau. Prenons par exemple le cas de « France » comme pays de provenance en tant que concept et la couche « LSTM » de notre réseau :



On voit donc que les activations sur la heatmap sont assez ternes. Ce qui veut dire qu'en moyenne, les activations des films dont le pays de provenance est la France tendent à se situer dans l'intervalle [-0.5 ; 0.5] environ. Ce qui est logique puisque dans les films français, certaines activations doivent tendre vers 1 et d'autres vers -1, et on se retrouve donc avec des activations autour du milieu de notre palette de couleurs comme on peut le voir sur la légende à droite.

Ainsi, si on regarde les activations pour un seul individu appartenant au concept « provenant de France », on devrait voir une heatmap avec des couleurs bien moins ternes :



Ici, on voit la heatmap représentant les activations au niveau de la couche « LSTM » pour le film « Without Leaving an Address ».
On voit clairement que les activations sont plus prononcées et on remarque surtout que certains neurones comme par exemples les numéros 6, 20, 21, 26, 27 et d'autres sont activés de la même manière. Ceci pourrait être un indicateur de la présence d'un concept au sein même des activations du réseau de neurones.

LOGICIEL DE VISUALISATION

[intro]

Apprentissage de Qt

[bla bla]

Fonctionnalités

[bla bla]

Voir annexes

Boost des activations

[bla bla]

Conclusion

RESULTATS

COMPETENCES DEVELOPPEES

Glossaire

Intelligence artificielle symbolique : sorte d'IA se basant sur des règles / connaissances en imitant le raisonnement humain dans le but de produire des décisions.

Système à base de connaissance : programme utilisant une base de connaissances afin de résoudre des problèmes. Ce genre de système est en général composé d'au moins un moteur d'inférence et au moins une base de connaissances.

Annexes

[DESCRIPTION DE CHAQUE PAGE]

Remerciements

Je tiens à remercier chaleureusement M. Mathieu d'Aquin qui m'a permis de réaliser ma première expérience de recherche dans une entité que j'affectionne tout particulièrement.

Quatrième de couverture (changer titre)

- Mathieu D'Aquin : mathieu.daquin@loria.fr