Recomanador de pel·lícules utilitzant filtratge col·laboratiu amb xarxes neuronals

Treball final de l'assignatura Tècniques Avançades d'intel·ligència artificial.

Bernat Comas i Machuca
u1972813

Índex

1 Introducció i motivació

L'objectiu d'aquest treball és construïr un sistema de recomanació de pel·licules a usuaris. Es farà utilitzant el Dataset MovieLens — GroupLens i un algorisme basat en xarxes neuronals.

El motiu per el que he decidit triar aquest projecte és perquè soc un amant de les pel·lícules i com a tal m'he trobat sovint interaccionant amb motors de recomanació, tant de plataformes com Netflix o HBO com de pàgines com IMDB o Letterboxd. Sovint m'ha fet preguntar-me com funcionen aquest tipus de motors i aquesta és una oportunitat ideal per no només descobrir com funcionen sinó implementar-ne un jo.

Per fer-ho m'he estat documentant sobre mètodes de creació i entrenament d'aquests models, i finalment m'he decidit per utilitzar una tècnica anomenada Collaborative Filtering. Aquesta tècnica consisteix en trobar similituds entre els vectors d'entrada, en el nostre cas els de usuaris que miren pel·lícules i pel·lícules que son mirades per usuaris, per tal de fer recomanacions basades en la informació que sabem sobre persones que tenen gustos similars a ells. A més, es farà la implementació utilitzant xarxes neuronals amb l'ajut de la llibreria de python pytorch.

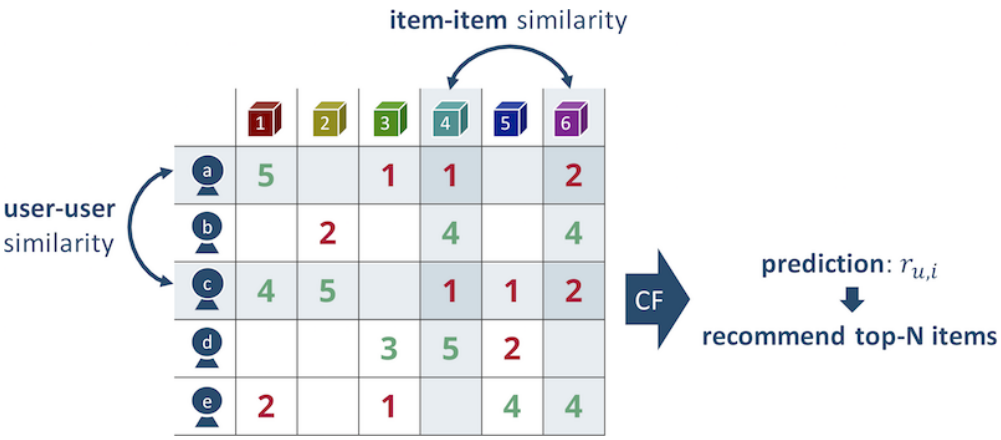

Figura 1: Esquema de Collaborative Filtering.

2 Fonts d'informació i coneixements dels que es disposa

Per poder dur a terme aquest treball s'ha utilitzat el dataset que ofereix MovieLens — GroupLens anomenat 25M Dataset, que conté 25 Milions de ratings que han posat usuaris a pel·licules, a més de molta altra informació que no s'utilitzarà en aquesta aproximació. Aquest dataset es pot trobar a GroupLens, Datasets[2].

Cercant a internet he pogut trobar diversos articles de persones que creen motors de recomanació utilitzant xarxes neuronals, com Bryan Chia (2020)[3]o Yun Liu et al. (2023)[4]. Aproximacions que utilitzin collaborative filtering també n'hem pogut trobar algunes, per exemple Benedikt Schifferer (2021)[5]o Xiangnan He, et al. (2017) [8]. Aquest últim ens mostra l'aproximació de collaborative filtering que incorpora la factorització de matrius a més del MLP, que serà finalment l'aproximació triada per dur a terme la pràctica.

Els beneficis que em donarà utilitzar collaborative filtering a través de xarxes neuronals, segons els diversos articles citats anteriorment, són: major precisió, capacitat per funcionar correctament incrementant el nombre de dades d'entrenament, robustesa als outliers i la gran flexibilitat que permeten. Tot i els nombrosos avantatges, diverses fonts ens recalquen que aquesta aproximació moltes vegades no arriba a superar un mètode naive que consisteix en recomanar a tots els usuaris els ítems amb més popularitat.

Les xarxes neuronals solen necessitar moltes dades per funcionar correctament. Per aquesta raó he triat el dataset que té el major nombre d'entrades: 25M de ratings de 62,000 pel·licules per 162,000 usuaris. Tot i això, com expliquem a continuació, no les farem servir totes.

Tot i que he trobat molts més exemples de persones que creaven les xarxes neuronals utilitzant Tensorflow Keras, m'he decantat per utilitzar la llibreria PyTorch per crear la xarxa neuronal seguint les recomanacions del professor.

3 Dades i Pre-processament

El dataset utilitzat, com hem explicat és el MovieLens 25M[2]. Dins aquest dataset hem utilitzat dues taules. En primer lloc la taula de ratings, que ens diu per cada nota posada, quin usuari la ha posat i a quina pel·lícula. En segon lloc la taula de movies, que ens diu per cada un dels identificadors de pel·lícula que trobem a la taula de ratings, quin és el seu nom. Aquesta última taula no pren paper en el sistema recomanador i és simplement un mitjà estètic per mostrar el nom de les pel·lícules recomanades.

El preprocessament ha consistit en dos passos:

En primer lloc hem filtrat les pel·lícules per tal de tenir un nombre més reduït d'instàncies. Això no és necessari i probablement ens donaria millors resultats si no ho féssim, però degut a les limitacions de la màquina en la que entrenarem el model, ens permetrà una millora significativa en la velocitat d'entrenament i de testeig. Filtrem totes les pel·lícules que han vist menys de 10000 usuaris i tots els usuaris que han vist menys de 1000 pel·lícules i ens quedem amb aproximadament un milió d'instàncies.

En segon lloc, com que volem un classificador binari que ens digui quina és la probabilitat de que a un usuari li agradi una pel·lícula, transformarem la columna rating en una columna interacció, que ens dirà si l'usuari hi ha interaccionat positivament (1) o no ho ha fet (0). Considerem que un usuari ha interaccionat positivament amb una pel·lícula si la seva nota és major a 3 sobre 5. Aquesta decisió també ens causarà una pèrdua significativa d'informació, però com que des de bon principi buscavem un recomanador binari, era el camí a seguir.

Com que estem construint un recomanador de pel·lícules, les pel·lícules amb les que l'usuari ha interaccionat negativament no ens interessen. Per aquesta raó i perquè a més ens permeten reduir la mida de les dades d'entrenament, eliminem totes les interaccions negatives i deixem únicament les positives. També eliminarem la columna de timestamp, ja que a efectes d'aquest treball no ens serveix.

Finalment doncs, ens queda una única taula que conté tres columnes. La primera són els identificadors dels usuaris als que recomanarem les pel·lícules. La segona és l'identificador de pel·lícula amb la que un usuari ha interaccionat en un moment. Finalment la última és una columna que només conté 1s si l'usuari ha interaccionat amb la pel·lícula en qüestió.

4 Arquitectura del sistema recomanador

En primer lloc cal dividir les instàncies en train i test. Utilitzem la funció train_test_split. Ens quedem amb un 80% per entrenar. Seguidament prepararem les classes que necessitarem per crear el nostre model. L'esquema que seguim per l'entrada és el que podem veure a la Figura 2. Tindrem per separat els vectors d'usuaris i d'ítems. Aquests els farem passar per diverses capes i finalment l'optimitzarem. La versió modificada d'aquest esquema, que serà la utilitzada per nosaltres es pot veure a la Figura 3.

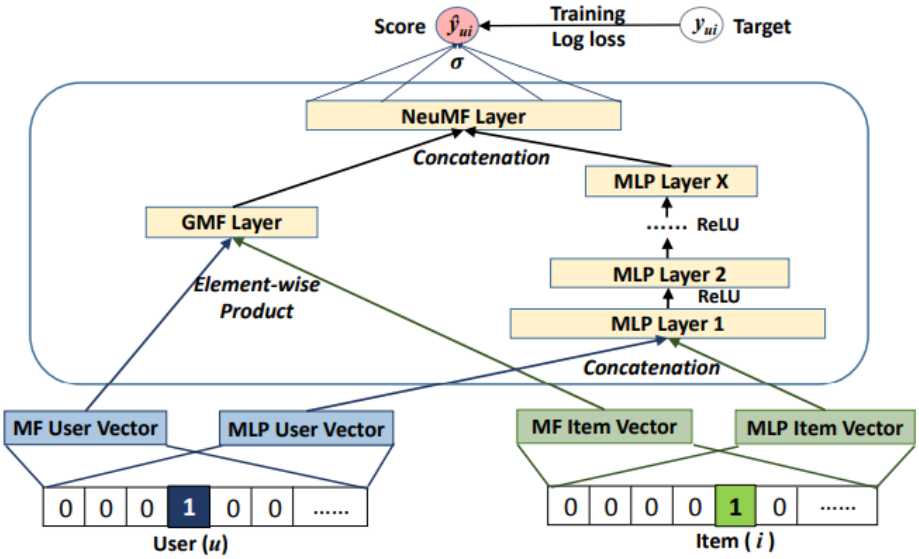

Figura 2: Arquitectura de la Xarxa Neuronal de Xiangnan He, et al. (2017).[8]

4.1 Xarxa Neuronal

La xarxa neuronal es troba definida a la classe NeuralCollaborativeFiltering. Aquesta classe conté el mètode d'inici, que declara les diferents capes que utilitzarem, i el mètode de forward, com és natural a totes les estructures de xarxa neuronal a PyTorch. Aquest mètode serà el cridat quan es passi al model les dades d'entrada i es demani una predicció. Conté el fluxe d'informació que composa la xarxa neuronal i es troba explicat a continuació i esquematitzat a la Figura 3.

En primer lloc separem els vectors de usuari i de pel·lícula de la taula de ratings. Aquests els posem en tensors de tipus Long a través del DataLoader explicat a 4.2.

L'entrada de la nostra xarxa neuronal seran els dos tensors de pel·licules i d'usuari. Aquests tensors els utilitzarem per crear per una banda la factorització de matrius (Xiangnan He, et al. (2017)[8]) i per l'altre el Multi-Layer Perceptron (MLP).

La Factorització de matrius consisteix simplement en multiplicar el tensor d'usuaris per el tensor de pel·lícules, per d'aquesta manera obtenir la matriu resultant i poder-la incorporar a la presa de decisions.

El tensor MLP l'obtenim concatenant els tensors d'usuaris i d'ítems i fent-los passar per un seguit de capes fully-connected. Aquestes capes tenen dimensions 32 i 16 respectivament. La primera capa té com a dimensió d'entrada la mida de la concatenació dels dos tensors. Utilitzem com a funció d'activació una ReLU ja que ens permet evitar valors negatius (es transformen a 0.0) i és computacionalment eficient i ràpida de calcular.

Els tensors resultants de la factorització de matrius i del MLP els concatenarem i els farem passar per una última capa FC que resulta en una única neurona. Aquesta neurona la passarem per un sigmoide, que ens donarà la probabilitat final resultant d'interacció entre un usuari i una pel·lícula, entre 0 i 1.

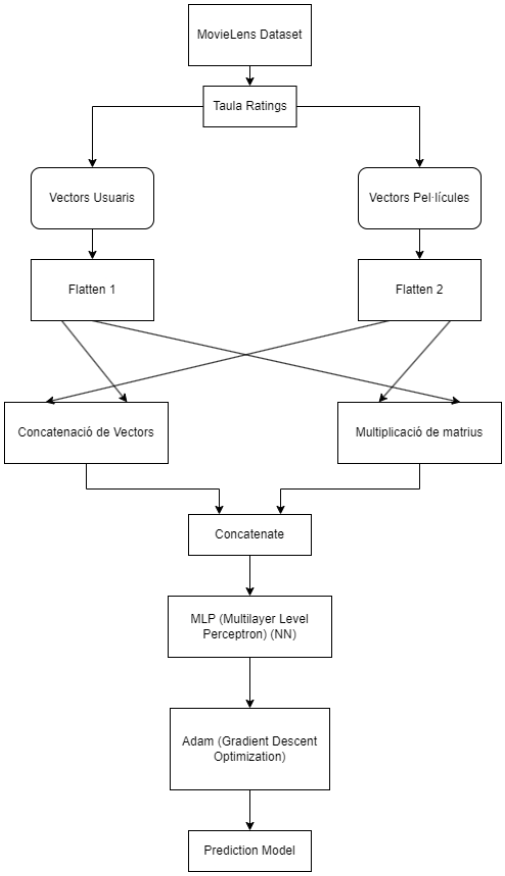

Figura 3: Arquitectura de la Xarxa Neuronal.

4.2 DataLoader

Ens interessa utilitzar un DataLoader que prengui i retorni tres paràmetres: usuaris, pel·licules i interaccions/prediccions. Per aquesta raó ens cal definir la nostra pròpia classe, que heretarà de la classe Dataset de Pytorch.

Aquesta classe només tindrà sobreescrits tres mètodes. En primer lloc el constructor passarà a rebre tres paràmetres: Usuaris, Pel·lícules i interaccions. Aquests tres tensors tindran la mateixa mida, així que si pregunten per la mida retornarem la de qualsevol dels tres. Quan ens preguntin per un item en particular retornarem una tripleta format per l'usuari, la pel·lícula i la interacció d'aquests dos.

Necessitarem instanciar dos DataLoaders, un per train i un altre per test. Aquests ens retornaran les dades dels seus respectius datasets i ho faran amb el batch size indicat.

5 Metodologia d'Entrenament

Per dur a terme l'entrenament del model utilitzarem l'optimitzador Adam. He triat Adam perquè és un model flexible i ràpid d'entrenar, i àmpliament utilitzat en models de recomanació.

La funció de pèrdua triada és la Binary Cross Entropy, àmpliament utilitzada per classificadors binaris. La fòrmula d'aquesta funció és la següent:

$$\ell(x, y) = -\frac{1}{N} \sum_{n=1}^{N} w_n \left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n) \right]$$

On N és el batch size, x_n la probabilitat predida per el sample n, y_n el valor real (que serà 1 o 0) i w_n és un pes opcional per cada element del batch.

Farem l'entrenament amb un learning rate i un weight decay de 0.001.

En l'entrenament no farem validació en test a cada pas ja que l'entrenament és llarg i tarda molt, i farem la validació en test una vegada acabat l'entrenament.

Altres temes a destacar de l'entrenament son els següents. Farem servir optimizer.zero_grad() per assegurar-nos que els gradients de cada batch es computen independentment, i el model està aprenent correctament. També s'utilitza la llibreria tqdm per mostrar el procés de càrrega.

Per tal de computar si una predicció ha estat correcte, en aquest cas es computa i mostra l'accuracy global. Tot i això, aquest accuracy no és el més adient per a la nostra aproximació. Això es deu a que com que estem fent un model recomanador no són iguals de rellevants totes les prediccions de pel·lícula, i únicament ens interessen els topk, és a dir les k pel·lícules que volem recomanar a l'usuari. Per aquesta raó ens centrarem en altres mètriques a l'hora d'evaluar el nostre model.

6 Metodologia de Testeig

El nostre procés de testeig consisteix en fer prediccions amb el model entrenat sobre dades que el model no havia vist anteriorment. D'aquestes prediccions n'extreurem l'accuracy, el mean squared error i el mean absolute error. Com hem explicat a l'apartat anterior aquestes mètriques no son les millors per evaluar el model, així que a continuació ens centrem amb les més importants: la Precision@K i el Recall@K. Nosaltres definirem top N com els N elements predits per un usuari que tenen la probabilitat d'interacció més alta.

La Precision@K es defineix com el nombre de recomanacions rellevants partit per el nombre de ítems recomanats. Per cada usuari, doncs, hem de calcular el nombre de ítems que hem recomanat correctament i el dividim pel total dels ítems que podriem haver recomanat (interaccio=1). Aquesta mètrica ens indica com de precises son les recomanacions en el top K d'ítems recomanats pel model.

El Recall@K es defineix com el nombre de recomanacions rellevants dividit pel nombre d'items rellevants. També per cada usuari haurem de calcular el nombre de ítems que hem recomanat correctament i dividir-ho pel total d'ítems que podriem haver recomanat (threshold major a un determinat). Aquesta mètrica ens indica quina proporció d'ítems rellevants es troben entre el top K de recomanacions.

La combinació d'aquestes dues mètriques ens permetrà evaluar les recomanacions que fariem a l'usuari final, ignorant així totes les prediccions que computariem però que no mostrariem a l'usuari perquè no es troben en el seu top K.

7 Resultats

Dividirem l'apartat de resultats en diferents subapartats per tal de mostrar no només els resultats de les top K recomanacions sinó també els resultats d'accuracy de train, test i la resta de mètriques que hem anat computant.

7.1 Resultats d'entrenament

Com hem dit anteriorment per mostrar el progrés d'entrenament hem utilitzat la llibreria tqdm. L'entrenament l'hem fet amb les següents dades:

batch size: 8192

epochs: 100

Finalment en acabat mostrem la training loss assolida, l'accuracy amb les dades d'entrenament i el temps que ha tardat en fer-se l'entrenament. Un exemple d'excució és el que es pot veure a la Figura 4

```
training loss: 47.170, training accuracy: 76.27%
CPU times: total: 5min 29s
Wall time: 16min 59s
```

Figura 4: Resultats d'entrenament del model.

Posteriorment hem mostrat gràficament l'evolució de l'accuracy i de la loss al llarg del procés d'entrenament amb 100 epochs. La Figura 4 ens mostra com ha evolucionat la pèrdua o loss calculada per la funció BCELoss. Com és normal la Loss disminueix al pincipi molt acceleradament, i a mesura que s'acosta al mínim ho fa amb menys acceleració.

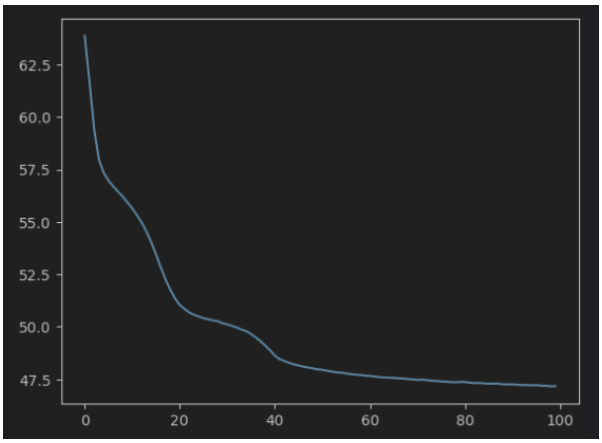

Figura 5: Evolució de la Loss al llarg de l'entrenament.

També hem pogut representar gràficament els resultats d'accuracy del model a través dels 100 epochs. A la Figura 6 podem veure com ha anat evolucionant l'accuracy, incrementant molt ràpidament inicialment i arribant a un punt estable al voltant del 76%.

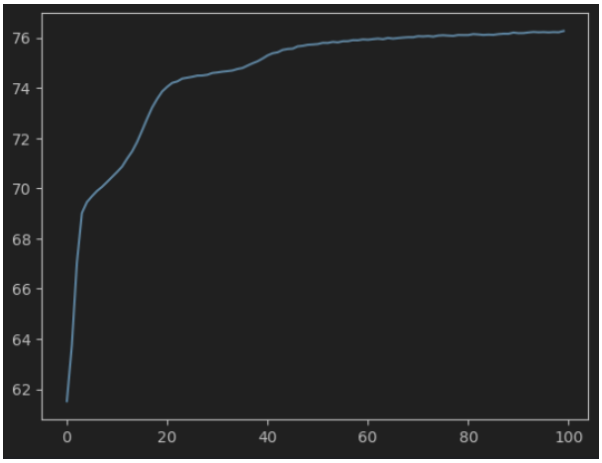

Figura 6: Evolució de l'Accuracy al llarg de l'entrenament.

7.2 Resultats generals de testeig

Els resultats generals de testeig es refereixen a les mètriques generals que hem pogut obtenir del testeig, que inclouen totes les prediccions, siguin positives o negatives. Podem veure els resultats de testeig amb el model explicat a la Figura 7.

```
Test accuracy: 0.7584%
Test MSE: 0.163
Test MAE: 0.330
```

Figura 7: Resultats de testeig.

7.3 Resultats de les predccions Top K

Una vegada testejat el model haurem de calcular en primer lloc el top K recomanacions per cada usuari per poder-ne extreure mètriques. Hem utilitzat una K de 5 pel·lícules, que seràn les que recomanarem a l'usuari.

Seguidament seguint la metodologia explicada a 6, calculem la Mean Precision entre tots els usuaris i el Mean Recall, així com les seves desviacions estàndard, per tal de comprovar si hi ha hagut una gran variabilitat entre les precisions per tots els usuaris.

Els resultats els podem veure a la Figura 7. La precisió at K ens ha quedat moderadament alta, amb un 87.64%. Tot i això la desviació estàndard també ens ha quedat bastant alta amb un 20.18%. Això pot ser degut a que hem hagut d'entrenar amb un nombre més reduït de pel·lícules i això fa que sigui molt precís per als usuaris amb gustos més normals però no tingui tanta robustesa als outliers i tingui poca consistència.

El recall del 10.32% ens indica que un 10% de les pel·licules rellevants es troben en el topK. Considerant que hem utilitzat una K de 5 pel·licules és un resultat acceptable. La desviació estàndard del recall és baixa, d'un 4.43%, el que vol dir que és generalment consistent.

```
Mean Precision@5: 0.8764
Std Precision@5: 0.2018
Mean Recall@5: 0.1032
Std Recall@5: 0.0443
```

Figura 8: Resultats de testeig dels top K elements.

7.4 Resultats estilitzats

Per tal de mostrar resultats més comprensius a l'ull humà he creat una petita funció que permet visualitzar per els diferents usuaris de test, quines eren les seves 5 pel·lícules amb més nota inicialment i quines han estat les que han aparegut al TopK. No es poden treure conclusions d'aquests resultats ja que no hem entrenat el model amb rating, sinó amb interacció (hem considerat igual un rating de 3.5 que un de 5) o no tenim en compte l'empat entre pel·lícules entre moltes altres raons.

La única raó per la que s'ha fet aquest apartat és perquè tots els altres estaven completament desconectats de les pel·lícules i es mostraven simplement les dades de resultat i he trobat interessant mostrar títols de pel·lícules encara que fos només com a curiositat.

Els resultats per dos usuaris qualsevols per aquesta mètrica explicada anteriorment es poden veure a la Figura 9

```
USER: 86930
Millors notes: ['Rush Hour 3 (2007)', 'Crimson Peak (2015)', 'Stepmom (1998)', 'Patch Adams
(1998)', 'Bend It Like Beckham (2002)']
Interaccions: ['Sixth Sense, The (1999)', 'Pirates of the Caribbean: The Curse of the Black
Pearl (2003)', 'Taken (2008)', 'Bourne Identity, The (2002)', 'Forrest Gump (1994)']

USER: 111073
Millors notes: ['Usual Suspects, The (1995)', 'Prestige, The (2006)', "Adam's Apples (Adams
æbler) (2005)", 'Rushmore (1998)', 'American Beauty (1999)']
Interaccions: ['Goodfellas (1990)', 'Eternal Sunshine of the Spotless Mind (2004)', 'No
Country for Old Men (2007)', 'Moon (2009)', 'L.A. Confidential (1997)']
```

Figura 9: Resultats de pel·lícules recomanades per a dos usuaris aleatòris juntament amb les seves pel·lícules amb més nota.

8 Conclusions i línies futures

En aquest treball, hem vist com crear un recomanador de pel·licules utilitzant Neural Collaborative Filtering i utilitzant tant un MLP composat per diverses capes totalment connectades com també afegint la factorització de matrius. En quant a la qualitat del model, hem pogut millorar els resultats en quant a Precision@K respecte el paper que implementa la factorització de matrius[8], que tenia un accuracy de menys de 75%. També hem millorat aquesta mètrica respecte el video de dataroots[7], on s'intentava assolir el mateix objectiu.

Ha estat una pràctica molt enriquidora ja que ha estat la meva primera vegada treballant amb xarxes neuronals i he hagut de fer molta recerca per trobar informació sobre

com dur a terme el meu objectiu. Hi ha hagut molts moments en que m'he encallat i he perdut molt de temps en coses de poca importància però estic content perquè he après molt i aquest és un tema que m'interessa.

Com a apreciació personal, trobo que està molt bé que se'ns permeti fer un projecte lliure d'intel·ligència artificial ja que ens permet centrar-nos i aprendre més de l'apartat que més ens interessi. Tot i això probablement m'han faltat alguna hora on poder fer treball a classe amb ajuda del professor perquè com a mínim el meu és un tema molt dens i em va costar fer les primeres passes.

Pel que fa a les línies futures d'aquest treball en concret, podria ser interessant crear una interfície de text on un usuari pugui entrar les pel·lícules que ha vist i que el recomanador retorni el top K pel·lícules que li podrien agradar.

Des d'un punt de vista més de ML podria ser molt interessant ampliar el focus de la xarxa neuronal i poder incorporar més dades per intentar fer les prediccions més precises. Per exemple d'aquest mateix Dataset es podrien afegir els gèneres de les pel·lícules, per tal de recomanar gèneres que et solen agradar o bé el timestamp, per tal que faci recomanacions tenint en compte el moment del dia en que l'usuari vol veure una pel·lícula o fins i tot tenint en compte l'any en el que ens trobem i el canvi de gustos que ha tingut l'usuari. Fins i tot aquest Dataset també ens ofereix els tags que han posat els usuaris a les diferents pel·lícules, de manera que es podrien fer moltes aproximacions diferents tenint sempre com a objectiu el model recomanador.

9 Bibliografia

[1] GroupLens, *Datasets, MovieLens* (Enllaç a la pàgina)

[2] GroupLens (2019), *Datasets, MovieLens 25M* (Enllaç a la pàgina)

[3] Bryan Chia (2020), *Using A Neural Network to Recommend Shows on Netflix:* (Enllaç a la pàgina)

[4] Yun Liu et al. (2023), *Knowledge-aware attentional neural network for review-based movie recommendation with explanations* (Enllaç a la pàgina)

[5] Benedikt Schifferer (2021), *Using Neural Networks for Your Recommender System* (Enllaç a la pàgina)

[6] Christina Ellis (2022), *When to use neural networks* (Enllaç a la pàgina)

[7] dataroots, (2020), *Neural Recommender Systems [Video], Youtube* (Enllaç a la pàgina)

[8] Xiangnan He, et al. (2017), *Neural Collaborative Filtering* (Enllaç a la pàgina)

[9] Singh, A. P. (2021), *Steps you should follow TO Successfully Train MLP - Analytics Vidhya. Medium.* (Enllaç a la pàgina)

[10] rec-tutorials (2021), *Neural Matrix Factorization from scratch in PyTorch, GitHub* (Enllaç a la pàgina)

[11] Kelsey Mays (2016), *Recall Basics: Everything You Need to Know, cars.com* (Enllaç a la pàgina)

[12] Sonu Airen, Jitendra Agrawal (2022), *Movie Recommender System Using K-Nearest Neighbors Variants - National Academy Science Letters* (Enllaç a la pàgina)

[13] Konstantin Rink (2023), *Mean Average Precision at K (MAP@K) clearly explained* (Enllaç a la pàgina)

[14] Maher Malaeb (2017), *Recall and Precision at k for Recommender Systems* (Enllaç a la pàgina)

[15] yoongi0428 (2021), *Recommender System in PyTorch* (Enllaç a la pàgina)

[16] PyTorch Contributors (2023), *Pytorch Documentation* (Enllaç a la pàgina)

[17] PyTorch Lightning, *Validate and test a model (Intermediate), PyTorch Lightning* (Enllaç a la pàgina)