Recommender Systems: Using Neural Networks for Content-Based and Collaborative Filtering

wrjx35
Department of Computer Science
Durham University
Durham, UK
wrjx35@durham.ac.uk

***Abstract*—Recommender Systems have become used in every day life to help suggest content to users across a range of domains. Two of the most common methods used to do this are Content Based Filtering and Collaborative Filtering. This paper proposes the use of Neural Networks within each of these techniques in order to create useful recommender systems within the movie recommendation domain.**

I. Introduction

Today there are an abundance of online streaming services such as Netflix, Amazon Prime, Disney+, Hulu, Roku etc. Each of these services provide a large amount of content to their respective users in order to stay competitive with one another within the market. Whilst this is a win for consumers in the sense that they get good value for their money, without some way to filter through the available content, it can leave them with an overload of choice when deciding what to watch.The use of recommender systems in this domain allows for the services to provide carefully tailored suggestions to their users and reduce the effects of choice overload.This paper aims to demonstrate two such recommender systems which both at their core make use of neural networks. The first model will make use of the Collaborative Filtering based model NeuMF proposed by Xiangnan et al. [ref] to suggest films to users based on the viewing histories of users with similar interests to themselves. The second model will be a Content Based Filtering system which makes use of Google's Word2Vec model [ref] in order to generate a user profile vector based on the metadata of films they have already watched. This user vector will then be compared against vectors generated for the individual films to suggest films which closely match the users taste.

II. Methods

A. Neural Matrix Factorisation

The implementation of Neural Matrix Factorisation (NeuMF) [2] for this project was trained on the MovieLens 25M dataset (downloadable at https://grouplens.org/datasets/movielens/25m/). This data contains 25 million ratings made by 162,000 users across 62,000 different movies. The ratings provided in this dataset give an example of explicit feed back (i.e. the user is telling you exactly what they think of the content), however NeuMF is designed to work with implicit feedback (feedback derived from whether a user interacted with an object or not) and so the ratings needed to be converted into implicit feedback by converting them all to binary values. In this instance all items have a rating and so must have been interacted with by the user, therefore they are given a value of one. As the dataset now only contains items with positive instances, for each positive item in the dataset, the user is also assigned 4 items with a value of 0 (negative instances) to stop the model always predicting the class label of 1 (the choice of 4 negative instances was selected as per the authors GitHub implementation of the model). Whilst the NeuMF model could potentially be adapted to use explicit feedback (by turning it into a regression model to predict how a user would rate each film) which could lead to suggestions which better reflect the users tastes, given the domain of the problem, implicit feedback seems more appropriate, as services like Netflix don't give their users the opportunity to rate films explicitly.

The NeuMF model itself works by combining a Matrix Factorization model and a Multi-Layered Perceptron.

Matrix factorisation [3] is an older recommendation model which takes one-hot encodings of user-item (in this case user-film) pairs and generates embeddings from them both, represented as a latent vector. The dot product of these vectors is then calculated to represent the predicted class. On its own matrix factorisation is not powerful enough to accurately learn the relationships between users and items, especially on sparse matrices and on particularly large datasets such as MovieLens 25M. This is why the combination with the multi-layered perceptron is necessary.

The multi-layered perceptron aims to learn the relationship between the users and the films latent vectors through the use of a deep neural network. The use of a deep neural network allows for the deeper, non-linear relationships between the inputs to be learned, which the matrix factorisation model is not advanced enough to learn on its own. The MLP model is built up of linear layers with ReLU activation functions.

The NeuMF model (shown in Fig. 1) then takes the concatenation of the outputs from the MF and MLP models and passes them through a dense linear layer followed by a sigmoid activation function to output the likelihood that a positive class is found (i.e. that the user would interact with the item).

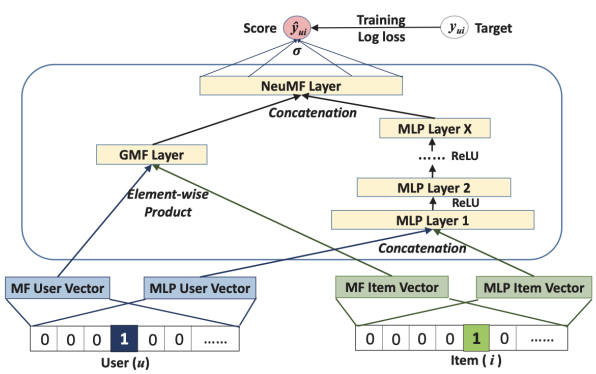

Fig. 1. Depiction of the NeuMF architecture taken from [2]

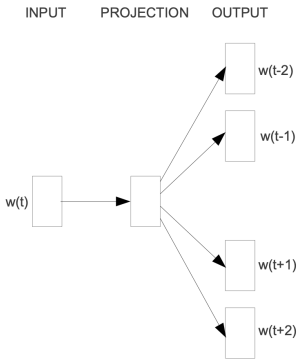

Fig. 2. Depiction of the skip-gram architecture taken from [4]

The model itself was trained over 200 epochs using Adam optimisation, a batch size of 1024 and binary cross-entropy loss.

B. Word2Vec Content Based Filtering

Like NeuMF, this model also uses the MovieLens 25m dataset to provide interactions between users and different films. In addition to this, the metadata and credits data from The Movies Dataset (downloadable at https://www.kaggle.com/rounakbanik/the-movies-dataset) this contains a vast array of metadata and credits lists for 45,000 of the movies in the MovieLens Dataset. In this model, only movies where there is metadata and credits lists available are used.

For each film used, the metadata and cast list was used to generate a description of the film using the names of the 3 main cast members (ranked according to their order on the credit list), the name of the director(s), up to 3 of the genres used to describe the film and up to 3 production companies involved in the making of the films. The directors and production companies were weighted double that of the cast and genres, as it can be argued that these give more weight to the style of the film (i.e. people who enjoy Pulp Fiction are likely to enjoy other Quentin Tarantino films and people who watched Iron Man are likely to want to watch other MCU films). These descriptions are then passed to the Word2Vec model. This method of building the movie descriptions is inspired by work from [1].

The Word2Vec model [4] uses a skip-gram approach to learn word embedding vectors based on the relationships between the surrounding words contained in a large corpus of text. In this case, the corpus is the list of descriptions of each film that were generated from the metadata. For example, the model should learn that there is a relationship between ”TomHanks” and ”TimAllen” as the both appear in the descriptions for films in the Toy Story series, and so their vectors should be similar to one another.

This is learned through the use of a log-linear classifier with continuous projection layer (as shown in Fig. 2) which takes as input a each word from the corpus and and tries to predict words within a certain range of the word.

The model was trained on the description of the films for 100 epochs. This resulted in vectorised word embeddings for each unique token in the list of film descriptions. In order to get a vectorised representation of each individual film, the average vector was computed from the embedding for each word in the description.

$$v_{film} = \frac{\sum_{word \ni desc_{film}} v_{word}}{|\text{desc}_{\text{film}}|} \tag{1}$$

The vectors generated for each film were then used to calculate the vectors to represent the users profile, by taking the average vector across the films that the user had watched.

$$v_{user} = \frac{\sum_{film \ni watched_{user}} v_{film}}{|\text{watched}_{\text{user}}|} \tag{2}$$

The user recommendations were then generated by finding the top 10 films whose vector most closely matched the vector of the individual user. This calculated using the cosine similarity between the user vector and the film vectors.

$$Top10_{user} = \underset{film \ni films}{max10}\ cosinesim(v_{user}, v_{film}) \tag{3}$$

III. Interface

The recommender systems created using the above methods first take as input a user ID number. They are then given the option to either view their recommendations or update the system by providing information about a new film they have watched. If they choose to view their recommendations the they are presented in a tabular format. Otherwise they are asked for the name of the film they have just watched. In the Word2Vec model as the user vectors and film description vectors are readily available, the user vector can be updated straight away to reflect the changes. The NeuMF model however cant do this as the model requires heavy computation due to the 25 million ratings used and the number of training iterations required. Therefore the new interaction is stored in a separate file which is combined with the training data next time the NeuMF model is trained.

IV. Evaluation

The performance of systems were evaluated using two different metrics. The first metric used was the Hit Rate @ 10 which measures how likely the system is to recommend items that the user will interact with. This is done using Leave-One-Out cross-validation, whereby one of the items interacted by each user is left out of the training set. For each user, the left out interacted item is then attached to a random sample of data which is fed into the model and used to generate predictions. Hit Rate is then calculated by averaging how often the interacted item is included in the Top N (in this case N = 10) recommendations.

$$HR@10 = \frac{|U^N_{hit}|}{|U_{all}|} \tag{4}$$

Measuring hit rate is important as it shows how accurate the systems are at suggesting items the user will like. However a low hit rate does not necessarily indicate a bad recommender system as the items suggested may still be relevant to a users tastes even if the cross-validation item is not included in the top 10.

The second metric used was a measure of diversity; i.e. how different the recommendations produced by the system are from one another. This was calculating by taking the top 10 recommendations for a sample of 1000 users and taking the average cosine similarity between each recommended film using the vectors generated by the Word2Vec based model.

$$diversity = \sum_{U \ni U_{all}} (1 - similarity(Top10_U)) \tag{5}$$

Measuring the diversity of the recommendations is important as it gives an idea of how well the system allows for the user to explore the available data. For example, a model with low diversity may only suggest a very specific type of film based on the users interaction history (e.g. suggesting only James Bond films because the user had watch Casino Royale). Whilst these recommendations may align with the users interests, the system may be over-fitting to a too specific criteria and therefore filtering out content that the user might have enjoyed. On the other hand, a model with very high diversity could be achieved by just randomly suggesting films. This is obviously not useful in this domain as the recommendations are going to be too varied to fit to a users interests. Therefore a good recommender system will have a balanced level of diversity amongst its recommendations.

	HR@10	Diversity
NeuMF	0.59	0.59
W2V	0.30	0.40

Fig. 3. Hit Rate @ 10 and Diversity scores for each system

Figure 3 depicts how well each system performs under the required metrics. NeuMF shows a significantly better hit rate, suggesting that its recommendations are more accurate to the users profile. However, this makes sense given the use of a deep neural network to find underlying patterns in the distribution of the user interactions. Both models show a moderate level of diversity, which indicates both that the design of the systems is working correctly, i.e. the recommendations are bound by some criteria (the user profiles) and that they are not over-fitting.

V. Conclusions

This paper has shown that it is possible to use neural networks to create content-based and collaborative filtering recommender systems that produce accurate and suitably diverse recommendations with in the film recommendation domain. Whilst Neural Matrix Factorisation appears to outperform the Word2Vec content-based filtering system, the performance of NeuMF could be further enhanced by training the model with weights learned from pre-training individual GMF and MLP models. Future work could also look to combine these methods to create a hybridised recommendation model that may produce even better recommendations.

References

[1] Hung-Wei Chen, Yi-Leh Wu, Maw-Kae Hor, and Cheng-Yuan Tang. Fully content-based movie recommender system with feature extraction using neural network. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, volume 2, pages 504–509, 2017.

[2] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[3] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the First International Conference on Learning Representations*, 2013.