FIT 5212 - Data Analysis For Semi Structured Data
Assignment - 2

Name: Anoop John
ID: 31084354

PART 1 - Recommender Systems

Introduction

Social media platforms are a rich source of semi-structured data. We have been provided with datasets collected from an online photo-sharing social media platform known as Flickr. In this platform, people share photos with others and make connections. As our task is concerned, we will be using different algorithms to build a recommendation system which can be used to recommend items (photos) to each user using the data we have.

Before talking about the model building process, we will look into different types of algorithms used for recommendation systems. These are as follows:

1. Content - Based Recommender System: This type of recommendation system makes use of the user profile, as it learns the profile of the new user's interests based on the objects that the user has rated. Thus, this type of recommendation system recommends similar items to the users that they have liked in the past.
2. Collaborative Filtering - These are the most famous and widely used algorithms, as they are used to aggregate the ratings provided by users, and they can also find commonalities among users on the basis of their ratings. The main advantage of collaborative filtering techniques is that they are not dependent on any machine-readable representations of the items being recommended and they seem to work exceptionally well in complex scenarios where the variations in taste influences the variation in users preferences.
3. Hybrid Recommender Systems: Combining one or more algorithms results in hybrid recommendation systems. There are different ways in which we can build hybrid systems. One example of a hybrid recommendation system, is a Hybrid Weighted Recommender, where the results obtained is from the combination of content based and collaborative filtering, where the weights for each system is initialized to be the same, but gradually changes based on the users ratings or preferences.

We will be building 2 types of Collaborative Filtering models as given below:

1. Matrix Factorization model using Alternating Least Scores (ALS)
2. Logistic Matrix Factorization model (LMF)
3. Bayesian Personalized Ranking model (BPR)
4. Neural Collaborative Filtering (combining matrix factorization & Multi-layer perceptron)

In our case, we will consider the rating for an item as a case of implicit feedback, where if the rating is '1', it means the user liked the item, else be the case that the user has not interacted with the item or they did not like the item.

We will briefly cover each model in the coming sections, and at the end we will summarize our findings by providing the results obtained from our analysis.

Matrix Factorization model using Alternating Least Squares (ALS)

Matrix Factorization is a method to generate latent variables when multiplying 2 different sparse matrices. In Collaborative Filtering, this is exactly what we do, as in, we take the users and items matrices, and we tend to identify the relationship between the users and items. For our task, using the inputs of the users rating for each item, we would like to recommend new items to the user from a candidate list of items.

The dot product of user and item matrix can generate the rating matrix, while the user matrix is the shape of k (users) * f (features) and the item matrix is the shape of j(items) * f (features). From user's and item's matrices, we can identify the relationship where in our case, it is whether the user has liked the photo or not.

Alternating Least Squares **(ALS)** is a matrix factorization algorithm that can be ideally used for large-scale Collaborative Filtering. ALS has done a good job in solving scalability problems and also the sparseness of the rating/interaction data, and as its implementation is quite straightforward, it scales well for huge datasets.

Characteristics of ALS algorithm is given as follows:
1. It uses L2-regularization to optimise the loss function
2. ALS minimizes two loss functions alternatively; It first holds the user matrix fixed and runs gradient descent with the item matrix, and vice versa.

Before building the model, we have used alpha (α) as a constant value for the rate of increase in confidence for any user-item pair, as per given in the research paper, which is provided in the reference section.

We have considered alpha, factors of the latent variables, regularization parameter as the hyper-parameters for our model with the same iteration value. To validate the results, we have used the Normalized Discounted Cumulative Gain **(NDCG)** as an evaluation metric to check the performance of our ALS model for different values of the hyperparameters. NDCG is a measure for ranking quality, and it can be defined as the ratio of the user's DCG score over the ideal rankings DCG score. We have used the validation dataset to perform hyper parameter tuning.

The best values for the hyper parameters are given as follows:

1. Alpha (α) = 50
2. Factors of the latent variables = 8
3. Regularization = 0.1
4. Iterations = 100

We obtained the best values for our hyperparameters and have used these values to build our ALS model. Using our model, we have got the recommended list of items for each user. We then sort this list and use it to get the top 15 recommendations based on the test data.

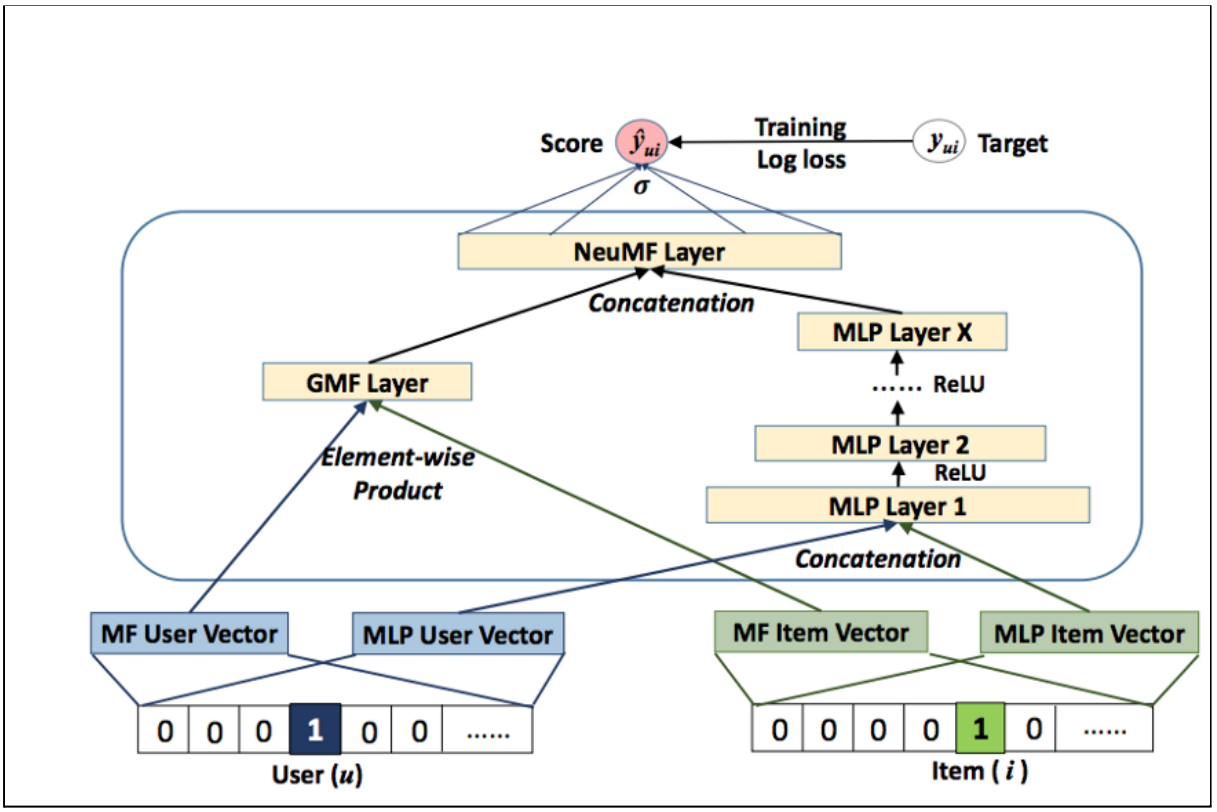
Other Implicit Models

Using the implicit package we have built Logistic Matrix Factorization mode, and Bayesian Personalized Ranking model. Logistic Matrix Factorization is a collaborative filtering algorithm that uses user-item interaction to build the probability distribution to validate if a user likes an item or not. Whereas, Bayesian Personalized Ranking uses the likelihood function for each user-item and also the posterior probability to decide whether the user will like the item or not.

Neural Collaborative Filtering

There is one limitation in using matrix factorization for collaborative filtering, as the simple multiple or say, the dot product of latent features might sometimes not help us in getting the complex structure of the user-item interaction. Neural Collaborative Filtering (NCF) could prove helpful in such scenarios. In general, NCF can help us model a better interaction function to represent the latent feature interaction for the users and items.
Characteristics of Neural Collaborative Filtering are as follows:

1. It uses a neural network architecture to model the user-item feature interaction. It mainly uses a Multi-Layer Perceptron (MLP) to learn the user-item interactions. This can be considered as an upgrade over the matrix factorization as a multi-layer perceptron can learn any continuous function and also has a high degree of nonlinearities due to the presence of multiple layers, enabling it to be well suited to learn user-item interaction function.
2. This technique also generalizes and expresses matrix factorization as a special case of Neural Collaborative Filtering, as matrix factorization is a renowned algorithm for building recommendation systems.



From the above figure, we can see that Neural Collaborative Filtering (NCF) consists of 2 core components, which is the Generalized Matrix Factorization and the Multi-layer Perceptron. It is important to note that both these components have separate item and user embeddings, as it can optimize the embeddings such that it can learn the user-item interaction better. Due to this structure, this model has several advantages, which are as follows:

1. The Generalized Matrix Factorization (GMF) applies the linear kernel to model the user-item interactions.
2. Multi-layer perceptron (MLP) uses several neural layers to model the non-linear user-iterm interactions.

NCF uses the sigmoid activation function for the GMF output as it can help the model to learn the edge weights from the data using the log loss function, whereas ReLU is the activation function used for the MLP.
The backbone of NCF is considered as the NeuMF layer as shown in the figure, where the outputs from the GMF and MLP are concatenated and used as the input to this layer.

We have built the NCF model in the notebook using the keras & tensorFlow library. In order to perform modelling in tensorFlow, we need to convert a pandas dataframe into a tensorFlow dataframe. After doing so, we have fitted our NCF model to the dataset, and have trained the model for 5 epochs. Using this model, we have made predictions on the test set and have saved those predictions to a dataframe which was then uploaded to kaggle. The results of both the algorithms are provided in the next section.

Results

We have successfully built 4 different models to recommend items to the users. From our analysis, we identified that the model built using the Alternating Least Squares algorithm outperformed the other 3 models. A possible reason for this outcome would be the fact that in ALS we minimize the entire loss function at once, by adjusting half the parameters, as the loss function is quadratic in nature and has an easy solution, when half the parameters are fixed. Hence, due to this there is no gradient in each optimization step as the problem is convex.

The performance of each model in terms of the NDCG scores for the test data are as follows:

Alternating Least Squares (0.22) > NCF (0.11) > Logistic Matrix Factorization (0.10) > Bayesian Personalized Ranking (0.06)

PART 2 - Node Clustering

Introduction

In this section, we are assigned with a task of performing node clustering of articles based on their content and other graph information. Clustering is basically a type of unsupervised learning where objects/data points with similar features will appear as part of a certain cluster. The information required for the task has been segregated into 3 different text files. We have read these files into our notebook and have extracted all the article information such as article ID, title and their respective label, and created nodes for our graph, using the **networkx** package. In addition to this, we have added edges to each article, as these edges denote that there is a relationship between 2 articles. In some cases, few articles were self independent, as they did not have a title or a label, so we have excluded them from our analysis, as we believe they do not have a significant impact on our task. After completion of the node and edge creation, we have a graph with all the relevant articles.

Analysis

We have performed 3 different types of clustering algorithms namely:

1. K-Means Clustering - This is the simplest clustering algorithm, where we can iteratively assign labels to the data points based on their features.
2. Agglomerative Clustering - The main idea behind agglomerative clustering is that each node starts in its own cluster, and recursively merges with the pair of clusters that minimally increases a given linkage distance.
3. Spectral Clustering : This is an algorithm primarily based on graph theory, where we can classify a node into a community based on the edges connecting them.

Using 2 different types of embeddings namely:

1. Word Embeddings - we have made word embeddings using Word2Vec to represent the title of each article.
2. Node Embeddings - we have made node embeddings using Node2Vec to represent the nodes of our graph.

We have also used a setting with no embeddings, and have gathered results of this setting, such that we have a baseline to evaluate the performance of clusterings with the 2 types of embeddings. Also, the metric that we have used to evaluate the performances of different clustering algorithms is the Normalized Mutual Information (**NMI**) score, which can be used to evaluate the quality of clustering.

In the case of word embeddings, we have retrieved the embeddings for each word and have added them with the other word embeddings to successfully replicate it as a sentence embedding.
As we know that most clustering algorithms are sensitive to initial values, we have used 10 different seed values to gather results. We have taken the average NMI score for each algorithm to validate our results.

Observations

The result of each algorithm using the word embeddings were significantly lower when kept in comparison with the baseline setup. The average NMI score for each algorithm was less than 0.1. This indicates that we needed more information and not just the titles to efficiently form the embeddings, where perhaps would have given a higher NMI score.

The node embeddings outperformed the latter 2 settings, as we got significantly higher scores for each algorithm. This technique proved effective as we believe we had enough information to represent these nodes as embeddings. In terms of NMI scores, the Agglomerative clustering performed relatively better than K-Means and significantly better than Spectral Clustering, as spectral clustering did not perform well since the affinity matrix was not a fully connected matrix.