FIT5212: Data Analysis For Semi Structured Data

Assignment 2

Recommender System Challenge & Node Clustering

Student Name: Udit Arora

Student Id: 31167799

Email: uaro0001@student.monash.edu

Kaggle Username: UditArora

Contents

Task 1: Recommender Systems

Libraries Used

- Pandas
- Numpy
- Scipy.sparse
- implicit

Data

- flickr_train_data.csv
- flickr_validation_data.csv
- flickr_test_data.csv

Description

The goal of this task is to recommend top 15 items for each user in the test data. The calculation is done with the interaction of the item and the user from the training and validation datasets. The task is performed using the implicit feedback method, the Alternating Least Squares algorithm. From this model we build the top 15 items recommendations from the given list of items for each user.

Building the model

After reading the data files, we build a sparse matrix by combining the training and validation data. It is a matrix with 0's and 1's with users represented by rows of the matrix and items by the columns of the matrix.

Following are the implicit method models that are built for this task.

	Model
ALS	Alternating Least Squares
LMF	Logistic Matrix Factorization

Many models were built for both of these methods and validated with NDCG scores obtained from Kaggle. The models were first built with default parameters and the result of the comparison was that the recommendations by LMF model were very low. Hence, the ALS method is chosen to build the recommender system.

Different sets of parameters were tried and tested before finalizing the parameters, the models were built with a hit and try method by increasing and decreasing values from an interval with a range of values for different parameters.

Parameters finalised:

- Factors = 5, Latent factors
- Iterations = 30, Iterations used to fit the data
- Regularization = 0.15, constant used for cost function
- Alpha = 40

The NDCG score for above parameters is 0.21706

The NDCG scores for different sets of parameters tried for the two models ALS and LMF are:

Model type	Alpha	Factors	Regularisation	Iterations	num_threads	Result	Model #
ALS	30	5	0.16	30	Default	0.20957	1
ALS	20	5	0.16	30	Default	0.20290	2
ALS	40	5	0.15	30	Default	0.21706	3
ALS	40	5	0.1	35	Default	0.21377	4
ALS	30	5	0.14	30	Default	0.20654	5
ALS	30	5	0.15	30	Default	0.20881	6
LMF	30	5	0.15	30	Default	0.08397	7

The model is built with the above mentioned features, for every user top 15 items are recommended from the given test data of 100 items for a user. The weight of a recommendation is stored as well with the recommended item.

	item_id	user_id	weight
0	8552.0	0.0	9.477407
1	3177.0	0.0	9.032385
2	740.0	0.0	7.520896
3	349.0	0.0	7.063329
4	6343.0	0.0	6.883848
...	...	...	...
51985	1873.0	3465.0	5.703547
51986	520.0	3465.0	5.692022
51987	1127.0	3465.0	5.576039
51988	6282.0	3465.0	5.532290
51989	56.0	3465.0	5.504817

For final Kaggle submission, the weight column is removed as per the specification and saved in a csv file.

Task 2: Node clustering in graphs

Libraries Used

- Networkx
- Node2vec
- Gensim
- Word2Vec
- TfidfVectorizer
- Stopwords
- Word_tokenize
- RegexpTokenizer
- WordNetLemmatizer
- LogisticRegression
- LinearSVC, SVC
- RandomForestClassifier
- Warnings
- Train_test_split
- Normalized_mutual_info_score

Data

- docs.txt
- labels.txt
- adjedges.txt

Graph Embedding

The graph is constructed using the data from the 'adjedges.txt' file which is read using the function read_adjlist() of the network library. After constructing the graph, the number of edges and nodes can be learnt from the object of the graph. The aim of this task is to classify the node based on node or embedding and justifying the method.

- **Node Embedding:** This is based on the Node2Vec algorithm**.** The Node2Vec algorithm is performed with parameters as 36 dimensions, number of walks 10 and walk length 10. After the transformation is done, the model vectorizes the input vector and gives an array of output label.

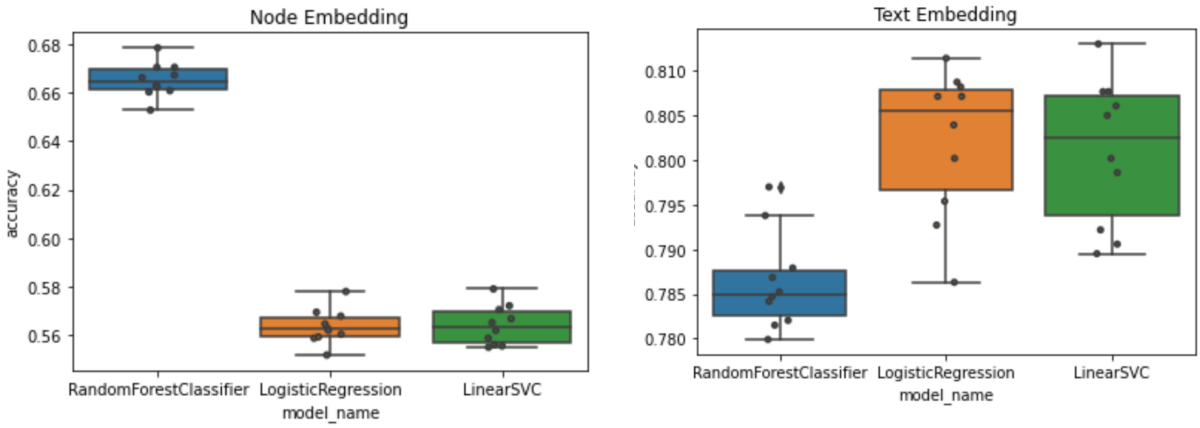
- **Text Embedding:** This can be performed using the following methods
    - **Word2Vec**: The vector generated here is from all of the topics, thus making the distinguishing factor less efficient, thus we can eliminate this method.
    - **TFIdf**: Here only the words are needed to classify the nodes, thus a regex tokenizer '[a-zA-Z]+' is used to build the model.

Building the model

Three different kinds of models are built for both node and text embedding using a 10 fold cross validation method.
Models considered are:

- RandomForestClassifier
- LogisticRegression
- LinearSVC

Following plots show the comparison of performance of the models for node and text embedding.



- For node embedding, RandomForestClassifier is the best performing model.
- For text embedding, LogisticRegression is the best performing model.

Comparing the models

The input data is divided into training and test data sets with test data being 30% of the total input data. We build the RandomForestClassifier using the vectorised node input and predicts the label for test data. A text embedded vectorised input data is used to build the logistic regression model and used to predict the label for test data.

We calculate the normalised mutual information(NMI) scores for the predictions of both the models and decide the better performing model

In [47]:

```
from sklearn.metrics.cluster import normalized_mutual_info_score
#calculate nmi scores for node embedding
print(modelname_rf)
print(normalized_mutual_info_score(node_y_test, node_y_predict))

#calculate nmi scores for text embedding
print(modelname_lr)
print(normalized_mutual_info_score(text_y_test, text_y_predict))
```

```
RandomForestClassifier
0.46570702871978714
LogisticRegression
0.5240933560107274
```

Conclusion

For the given network graph data, we can say that the text embedding model LogisticRegression performs the best as compared to the node embedding model.

Model	**Normalised Mutual Information score**
LogisticRegression, Text embedding	0.52
RandomForestClassifier, Node embedding	0.46

The RandomForestClassifier has low accuracy rate whereas in comparison the LogisticRegression has better accuracy rate of 0.52

A better model can be created if we can combine both text and node embedding methods and build a model.

References

- Monash FIT5212 Lecture Notes and Tutorials
- ALS model, https://implicit.readthedocs.io/en/latest/als.html
- LMF model, https://implicit.readthedocs.io/en/latest/lmf.html
- Graphs, https://networkx.github.io/documentation/networkx-1.10/tutorial/tutorial.html
- NMI score, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html
- Train test split, https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html