DEVELOPMENT OF RECOMMENDER SYSTEMS USING ML.NET
IZGRADNJA SISTEMA PREPORUKE POMOĆU ML.NET

Bahrudin Hrnjica, Denis Mušić, Softić Selver
Faculty of Engineering Sciences, University of Bihac, bahrudin.hrnjica@unbi.ba,
Faculty of Information Technology, University Dzemal Bijedic Mostar, denis.music@edu.fit.ba,
CAMPUS 02 University of Applied Sciences, Graz, Austria, firstname.lastbname@campus02.at

Keywords: machine learning, matrix factorization, ML.NET, product recommender

ABSTRACT

ML.NET is an open source framework developed by Microsoft for training, building and evaluation machine learning models. Recommender Systems are machine learning based algorithms. They found application in various business scenarios e.g. video on demand or music streaming like Netflix, YouTube, products sales recommendation such Amazon or content recommendation such as Facebook or Twitter. Besides applications used by multinational companies, the recommender systems found the application in small business like restaurant, supermarket, cinema, retail stores etc. In this paper the methodology and several use case scenarios were developed by using the ML.NET the machine learning framework built on .NET platform.

Ključne riječi: mašinsko učenje, matrica faktorizacije, ML.NET, sistemi preporuke

SAŽETAK

ML.NET je open source platforma razvijena od Microsofta za treniranje, izgradnju i evaluaciju modela baziranih na mašinskom učenju za .NET platformu. Sistemi preporuke predstavljaju algoritme bairane na mašinskom učenju. Oni nalaze primjenu u različitim poslovnim scenarijima poput video i audio streaminga poput usluga koje nude Netflix i YouTube, preporuke za prodaju proizvoda na online sistemima poput Amazona, ili preporuke sadržaja koju rade Facebook ili Twitter. Pored poslovnih aplikacija koje koriste multinacionalne kompanije, sistemi preporuke našli su primjenu i kod manjih poslovnih aplikacija poput preporuke proizvoda u kinima,restorana, supermakretima, i dr. U ovom radu prezentirana je metodologija izgradnje sistema preporuke, kao i nekoliko use case scenarija koji a koji se baziraju na korištenju ML.NET platforme.

1. INTRODUCTION

First works on recommender systems appeared in the early 90's when the internet and email started to be massively used and popularized. One of the first work about recommender systems was trying to reduce the amount of incoming documents due to increasing use of electronic mail [1]. The systems were able to filter the content based on the users' activities and collaborations. In the middle of 90's the first distributed recommended systems published on the network. For the first time the Group Lens company was performed automated collaborative filtering for the internet site *www.usenet.com* [2]. However, the Ringo systems implemented music albums and artists recommender [3], while the

Bellcore Video Recommender implemented the first online movie recommender [4]. In the beginning the used algorithms were based on users' inputs where similar users' behaviors classified by using "k-nearest neighbor" algorithm. In the first several years of 21st century research in recommender systems was intensified specially after Netflix announced the 1$ million Prize for a 10 percent improvement in prediction accuracy of the current recommendation system [9]. That was milestone in development of the recommender systems, since the competition succeeded to bring many developers and engineers together to work on improvements of the current recommended systems. In this time the Facebook and then little later Twitter became the main social stream, and everybody wanted to be on Facebook and Twitter to find their friends and followers. So, the recommended systems were improved in order to recommend similar friends by the same location, educations, and other properties.

2. RECOMMENDER SYSTEMS

The basic idea of the recommendation systems is based on the assumption that if the users share the same behaviors while buying in the online stores, selecting the similar artists, reading the same news, or watching the same movies, they will also have similar behaviors in the future. With other words, the recommender systems estimate users' preferences on items and recommended items, so that other users probably may like them. The recommended systems can analyze items on two ways: they analyze items in terms of how users like them, so that similar users will like same products, or analyze items in terms how items are popularized between users, regardless of the user's similarities. Based on the two fundamental concepts, the recommender systems can be classified on the main three types: content based, collaborative filtering, and hybrid-based systems [17,18].

Content based recommender systems analyze a set of descriptions of items previously rated by users and create a profile of a user behavior based of the attributes of the items rated by that user. For example, a user rates a book at online book store. The book has its attributes e.g. genre, title, author, type, prize. The attributes are used by the recommender systems in term that it can provide the recommendation list for the user.

Collaborative filtering based recommenders analyze users activities and try to find similarities between users, how they like, view or buy the same items. In that case users who shares the same behavior in the past, will probably have similar tastes in the future. For example, if the users 1, and user 2 have very similar purchase history, e.g. they liked items from the same category, bought the same items, and user 1 has recently bought a movie that user 2 has not yet watched., it would make sense to recommend the movie to user 2 as well.

Hybrid based recommender systems combine previous both techniques to generate better and more precise recommendations. For example, there is information about users purchasing history with details about individual items, and users. A recommender system could be enhanced by hybridizing collaborative or social filtering with content-based techniques, in order to overcome the problems identified from the previous types.

The formulation of the recommendation can be defined as *predicting the rating value* for a user-item combination or *prediction of the top k items* for a particular user.

The first formulation is more general and requires the user preferences over the set of items. For M users and N items the training data is provided as sparse matrix. The missing values are subject to prediction. The problem is also referred to as the *matrix completion problem*.

The recommender starts by defining a sparse rating matrix R with M users and N items. The goal is to create predicted dense interaction matrix $\hat{R}$ with all rating of users and items. Let the r_{ui} denote the preference of a user u to item i so that $\hat{r}_{ui}$ represents the prediction score. Since the sparse rating matrix R is not complete, two vectors are created.

The first vector (rows of R) $\boldsymbol{r^u}$ to represents each user:

$$\boldsymbol{r^u} = \{r^{u_1} \quad r^{u_2} \quad \dots \quad r^{u_{N-1}} \quad r^{u_N}\}, \tag{1}$$

And the second vector (columns of R) represent partially observed rating $\boldsymbol{r^i}$ to represent each item:

$$\boldsymbol{r^i} = \begin{Bmatrix} r^{i_1} \\ r^{i_2} \\ \vdots \\ r^{i_{M-1}} \\ r^{i_M} \end{Bmatrix}, \tag{2}$$

Since the matrix is sparse two sets are created: O and O^- denote the observed and unobserved interaction sets. Let k denote latent factor that represents latent dimension so that for two matrices $U \in \mathcal{R}^{M\times k}$ and $V \in \mathcal{R}^{k\times N}$ can be defined so that they satisfied:

$$\underbrace{\hat{R}}_{M\times N} = \underbrace{\boldsymbol{U}}_{M\times k} \times \underbrace{\boldsymbol{V}}_{k\times N}. \tag{3}$$

The prediction of the top *k* items put less general into more specific problem. The problem can be solved by calculating the first formulation and then rank the prediction of the top k-items.
The primary goal of recommender systems is increasing the sales and are utilized by merchants to increase their profit. The goal can be achieved by bringing relevant items to the attention of users, so that it increases the volume sales and profits for the merchant. On the other hand, recommendation can help users to improve overall satisfactory of the Web site. If the user receives relevant recommendation each time he visits the web site will be more satisfied with it and will use it again. In this way, the user will develop loyalty with the web site, but also for the merchandizing indicators of users needs, so the recommendation can be further improved. The recommender systems were not used only for sales. In case of Facebook or Twitter the recommendation is based on social connections, which have indirect benefits to the site by increasing its advertising profits.

3. ML.NET MACHINE LEARNING ON .NET PLATFORM

ML.NET is an open source framework developed by Microsoft for training, building, evaluation and deployment of machine learning models for .NET based applications [4]. It allows users to deliver custom machine learning models using C# or F# programming languages for various ML problems e.g. regression, classification, sentiment analysis, time series forecasting, recommendations. The framework was built on the pipeline pattern which is ideal for developing ML solutions. In the pipeline pattern, processing elements are arranged so that the output of each element is the input of the next. The idea was found in analogy to a physical pipeline [5]. In developing ML libraries, the pipeline pattern is proved to be very suitable, and it has been already used in similar .NET projects [6]. Since the central implementation is based on the pipeline pattern, very often ML models or ML projects are called ML pipelines [4].
When using ML.NET common ML project consist of predefined ordered steps called pipeline (Figure 1):

1. define a problem and collect the data related to the problem,
2. load the data into a `DataView` object,
3. define set of pipeline pattern based operations related for data transformation,
4. define set of features and label, for a ML algorithm,
5. train a model by calling `Fit()` on the pipeline,
6. perform set of statistical estimations for model evaluation,
7. in case the model is not satisfied the performance expectation, steps 3,4, 5 and 6 are repeated,
8. persist the model as file, for use in an application,
9. load the model into an `ITransformer` object for prediction,
10. make predictions by calling `CreatePredictionEngine.Predict()`.

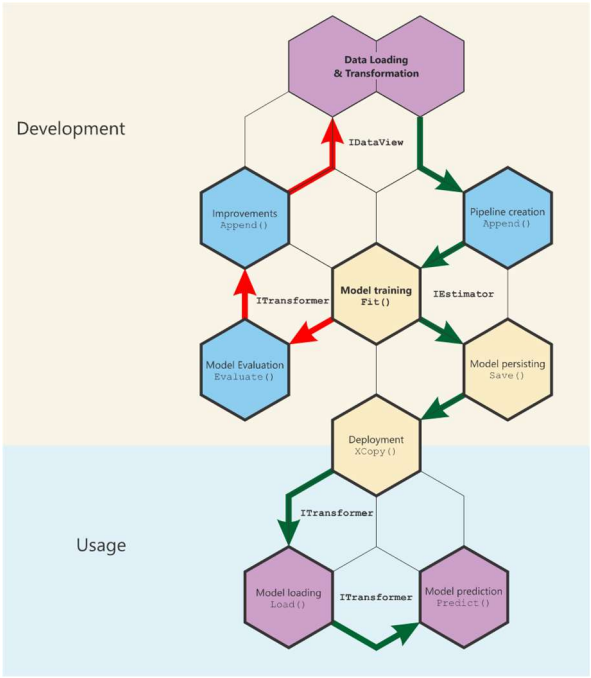

Figure 1: ML model lifecycle workflow

In ML.NET, central role of the data preparation is defined by the `IDataView` interface with implementation of the `DataView` class. The `DataView` class implements various methods used for data manipulation and preparation with abstractions similar with relational database. It provides compositional processing of schematized data with fully support of the pipeline pattern, with generics abstraction of primitive operators and the composition of multiple operators in order to achieve higher-level semantics such as the `FeaturizeText` transform. The operators implemented, based on the `IDataView` interface are able to gracefully and efficiently handle high-dimensional and large datasets with cursoring supports the well-known iterator model of databases [8].

4. BUILDING RECOMMENDER SYSTEM OF RESTAURANT DATASET

In this paper the recommender system was built for restaurant dataset using ML.NET. The dataset was created in the time period from Jul-2010 to Feb-2011. During that period users added and rated new and existing restaurants by filling the 21 attributes which 19 of them are provided by the user when he/she signs into the system. Data comprises 138 unique users that contributed with information about 130 restaurants and accumulated 1161 ratings. Possible rating values are 0, 1, and 2, where 0 indicates that the user does not like the restaurant, and 2 denotes a high preference [9]. The recommender systems try to predict user's rating for the unrated restaurants. In order to build recommendation engine, the selected algorithm was Field-aware Factorization Machines [10]. The algorithm uses Collaborative Filtering for recommendations which uses the fact that if a person A has the same opinion as a person B on the same restaurant, A is more likely to have B's opinion on a different restaurant than that of a random person.

4.1 Dataset preparation

Previously mentioned, the dataset collects the users rating for the restaurants in the specified time period. The dataset was divided in two sets: the *training set* which holds the 80% of the original data, and the *test set* which holds the remaining 20 % of the data. The user table contains 19 users attributes: `userID`, `latitude`, `longitude`, `smoker`, `drink_level`, `dress_preference`, `ambience`, `transport`, `marital_status`, `hijos`, `birth_year`, `interest`, `personality`, `religion`, `activity`, `color`, `weight`, `budget` and `height`. The restaurant dataset describes each restaurant with 23 attributes. Some of them are: `cuisine`, `alcohol`, `smoking`, `dress`, `accepts (type of payment)`,`parking`, etc. The values were generated by users from several possible options for each attribute. The whole datasets can be viewed from the official site at https://www.kaggle .com/uciml. The last dataset used in the recommender system is the rating table which collects the users, restaurant and user's rating for the specific restaurant. The rating table contains the rating values (0,1 and 2) of all rated restaurants. The rating table contains 1161 ratings. Since the rating values can take 0, 1 or 2 value, the whole rating column transformed into new binary column called `recommended`. The binary column indicates if the specific restaurant can be recommended for the specified user or not. Since the column values are mostly textual data, the data should be transformed into their numerical transformation by using several techniques which can be seen in the literature [7].

4.2 Model training and evaluation

Once the data is prepared the training phase consisted of the ML algorithm selection and calling the `Fit()` method by passing the transformed training dataset. The selected algorithm was Field- aware Factorization Machines, FFM which is proved to be very effective for building recommendation based on the multiple features [9]. Table 1 shows the parameters used during training of the recommender model.

Table 1 FFM parameters used in the paper

Parameter Name	Parameter Value
Learning Rate	0.013
Number of iterations	450
Shuffle	true
Extra Feature Columns	`"smoking_area","accessibility", "dress_code"`
Normalize Features	True

Table 2 Performance values for the restaurant recommender model

Parameter Name	Parameter Value
Accuracy	71.96%
Area Under Curve, AUC	82.10%
Area Under precision/recall Curve, ROC	74.82%
F1 Score	69.39%
Log Loss	0.82
Log Loss Reduction	0.18
Positive Precision	0.68
Positive Recall	0.71
Negative Precision	0.75
Negative Recall	72.88%

Once the model is trained and persisted on the disk, the model evaluation was performed by calculation of several performance parameters for test dataset. The model evaluation shown that total accuracy for the test data 73%, and the AUC=0.83, and ROC=74.5. The rest of the parameters are shown in Table 2. In case of random binary classifier, the ROC and AUC values are 50%, which is much lower than values of the recommender model presented in the paper.

5. CONCLUSION

The paper presents the methodology and use case scenario how to use ML.NET an open source framework in building recommend system for the restaurants. Through the several phases the recommender system was built by providing details on each phase. The paper present how to load, prepare and transform the data so that the feature analysis can produce the best possible ML model, by selecting only features which can significantly improve the model accuracy. Data transformation also included creation of the training and testing datasets, so that after training phase the model evaluation can provide reliable decision whether the model can be used in the production. Trough the example used in the paper, the model of the recommender system was successfully built using ML.NET platform and FFM algorithm. The evaluation shown the model have significant accuracy, which is far better than random model.

6. REFERENCES

[1] Goldberg, D. Nichols, D. Oki, B. M. Terry, D. (1992) Using collaborative filtering to weave an information tapestry, Communications of the ACM 35, no. 12, 61–70.

[2] Resnick, P. Iacovou, N. Suchak, M. Bergstrom, P. Riedl, J. (1994). ouplens: An open architecture for collaborative filtering of netnews. In: Proceedings ACM Conference on Computer-Supported Cooperative Work, pp. 175–186.

[3] Shardanand, U. Maes, P.(1995) Social Information Filtering:Algorithms for Automating "Word of Mouth". 1995. New York. ACM.

[4] Ahmed, Z., Amizadeh, S., Bilenko, M., Carr, R., Chin, W., Dekel, Y., Dupré, X., Eksarevskiy, V., Erhardt, E., Eseanu, C., Filipi, S., Finley, T., Goswami, A., Hoover, M., Inglis, S., Interlandi, M., Katzenberger, S., Kazmi, N., Krivosheev, G., Luferenko, P., Matantsev, I., Matusevych, S., Moradi, S., Nazirov, G., Ormont, J., Oshri, G., Pagnoni, A., Parmar, J., Roy, P., Shah, S., Siddiqui, M.Z., Weimer, M., Zahirazami, S., Zhu, Y. (2018), Machine Learning at Microsoft with ML .NET. ArXiv, abs/1905.05715.

[5] Vinoski, S. (2002), Chain of responsibility, IEEE Internet Computing. https://doi.org/10.1109/MIC.2002.1067742.

[6] Dobric, D., Hrnjica, B., (2017), LearningAPI, https://github.com/UniversityOfAppliedSciences Frankfurt/LearningApi, [accessed, july 2019].

[7] Hrnjica, B. Danandeh Mehr, A. (2019), Optimized Genetic Programming Applications: Emerging Research and Opportunities (pp. 1-310). Hershey, PA: IGI Global doi:10.4018/978-1-5225-6005-0.

[8] Feurer, M., Springenberg, J. T., Klein, A., Blum, M., Eggensperger, K., Hutter, F. (2015), Efficient and robust automated machine learning. NIPS.

[9] Vargas-Govea, B. González-Serna,G. Ponce-Medellín, R. (2011) Effects of relevant contextual features in the performance of a restaurant recommender system. In CEUR Workshop Proceedings.

[10] Juan, Y. Zhuang, Y. Chin, W.-S., Lin, C.-J. (2016) Field-aware Factorization Machines for CTR Prediction. https://doi.org/10.1145/2959100.2959134.