International Journal of Advance Research in Science and Engineering
Vol. No.6, Issue No. 04, April 2017
www.ijarse.com
IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

RECOMMENDATION SYSTEM FOR AN ANDROID LAUNCHER

Shreyans Jasoriya[1], Pranit Jaiswal[2], Abhinav Gautam[3], Mit Patel[4], Anand Pardeshi[5]

[1,2,3,4,5]*Information Technology, Fr. C. Rodrigues Institute of Technology, Navi Mumbai, India*

ABSTRACT

An android launcher completely changes the experience of using a mobile phone. It is the part of an android user interface that lets users customize the home screen, launch mobile applications, make phone calls, and perform other tasks. Currently, launchers offer mainly customization that provides cosmetic features and very few operational functionalities that are used on a daily basis. This launcher aims to create a user modeling system that develops unique customized user profiles based on their daily activities and offer them suggestions and smart notifications in the launcher itself based on the contextual data corresponding to tasks, interests, and habits.

Wide linear models and deep feedforward neural networks combine the benefits of both memorization and generalization to learn a unique user profile and provide recommendations, but they suffer from a poor initial performance due to lack of sufficient data. This paper proposes a new model that integrates the Wide & Deep model with a Bayesian linear hierarchical model that can help learn a basic user profile, by borrowing information from other user profiles.

Keywords: Android Launcher; Recommender Systems; Wide and Deep Learning Models; Bayesian Hierarchical Model

I. INTRODUCTION

In the current Android launchers, the ability to provide smart recommendations regarding the different types of utilities of an application remains unexplored in the mature development of Android launchers.

The proposed system is the recommender in a launcher that can offer cues for launching certain applications based on the user and contextual features like location, time and usage to enhance ease of use of the user interaction with the system. These features are used as a query to the recommender engine to provide personalized app recommendations in the app drawer of the launcher. The performance of the mobile operating system can be boosted by offering alternate solutions for installation/uninstallation of certain resource intensive applications by analyzing app usage statistics.

Our recommendation model exhibits both memorization and generalization features at the same time. In memorization, the items that occur frequently in the historical data are exploited to obtain correlation among them [1]. Transitivity of correlation and exploring of newly introduced feature combinations that has less likely been previously occurred in is known as generalization [1]. When building a recommendation system, a major

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

challenge is that the unique user profile for a particular user that is learned will be of low quality if for that user the amount of data is not sufficient [2]. This gives rise to the "cold start" problem, wherein a new user experiences poor initial performance until a dependable user profile is built [2]. To overcome the cold start problem, our model will implement Bayesian Hierarchical modeling approach which takes in the training sets of existing users and maps it onto new users to improve recommendation [3][4]. A basic training set for a new user is generated with the help of a predefined set of questions that a user must answer, and this training set is matched for similarities with the existing user profiles. The above process is carried out immediately after the installation of the launcher.

Using one-hot encoding, the sparse binary features are used to train logistics regressions [1]. When a user opens an app 'x' a binary feature "user_opened_app=x" has value 1 [1]. Cross-product transformations over sparse features are used to achieve memorization in an efficient way [1]. The value of query *AND(user_opened_app=x, impression_app=y")* is 1 when the user opens an app 'x' and then app 'y' is displayed next to it as a recommendation, in the app drawer of the launcher. This recommmendation is a result of frequent occurrence of two feature pairs that correlates with a target label [1]. Features that are less granular are used for adding generalization. Thus a query like *AND(user_opened_category=a, impression_category=b)* adds generalization but features must still be manually created as query-item feature pairs can't be generalized as long as they have not appeared in the training set [1].

We are going to present a new model that maked use of Wide & Deep models modified by integrating Bayesian Hierarchical Model to overcome poor initial performance in one model.

II.RELATED WORK

A. Wide & Deep Learning for Recommender Systems

A general recommender system takes in various user and contextual features to return impressions in response to a query [1]. This system records in logs all the queries and impressions as well as the user actions to use as training data for the model [1]. This is achieved by the usage of the wide linear model which is a logistic regression with cross-product feature transformations and feed-forward neural networks for developing a recommender system, is inspired from the Google Play Store Recommendation Engine [1]. Cross-product transformation of feature items make memorization practical and effective, while more feature engineering is required for effective generalization. Feed forward neural networks are used for generalization, as they don't require much feature engineering and they do this by using low-dimensional dense embeddings to generalize unseen feature item combinations. However, feed-forward neural networks when used alone can over-generalize and as a result start recommending less reliable feature items when the interactions between user and item are scarce and high-rank.

B. Efficient Bayesian Hierarchical User Modeling for Recommendation Systems

This paper examined the weakness of the EM-based learning approach for Bayesian hierarchical linear models and proposed a better learning algorithm called Modified EM [2]. The Bayesian hierarchical modeling approach is used in Content-based user profile learning that is a key to providing personal recommendations to the user. The model specializes in recommending items that have small number of ratings [2]. For building large scale recommendation systems, the concern for scalability of the system is addressed in this paper [2]. This paper

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

helped to make Bayesian hierarchical user models even more practical. The proposed new technique can further speed up the learning process of the model so that it can handle a huge system with hundreds of millions of users by adapting it to run simultaneously on a cluster of machines [2].

III. RECOMMENDER SYSTEM FRAMEWORK

This idea of adding generalization to linear models is inspired by factorization machines, [5] which is a combination of feature engineering generalization and factorization models superiority in the estimation of interaction between 2 variables as a dot product between 2 low-dimensional embedding vectors.

A. Proposed Models

1) Wide & Deep Learning

a. The Wide Linear Model

In this component, a wide linear model represented by $y = w^T x + b$ is present [1]. In figure 1, y is the prediction that is made, x = $[x_1, x_2, \ldots, x_d]$ is a vector of features having a range of d, w = $[w_1, w_2, \ldots, w_d]$ are the parameters of the model and the bias is given by b [1]. Raw input features and transformed features are included in the feature set. The cross-product is transformation is defined by the formula below:

=

Where c_{ki} has the value 1 if the i^{th} feature is a constituent of the k^{th} transformation φ_k, and 0 in all other cases [1].

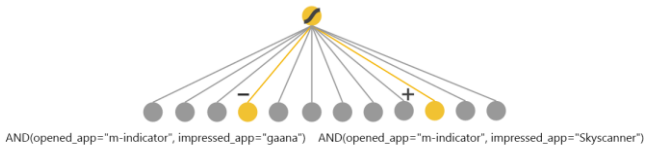

Figure 1: Wide Model. Adapted from " Wide & deep learning for Recommender systems", by H. Cheng et al, *arXiv preprint arXiv:1606.07792*. 2016

In our recommendation system the cross-product transformation for the query *AND(app_category=productivity, location="421301")* is 1 if and only if the constituent features *("app_category=productivity" and " location="421301")* are all 1, and 0 in all other cases [1]. In Figure 1, we have the query, opened_app = "m-indicator" and using memorization items that work best with this query are found [6]. The model predicts the probability of a user opening an app just after opening the app "m-indicator. The model memorizes that the feature *AND(opened_app = "m-indicator", impressed_app = "Skyscanner")* has a higher probability [6].

b. The Feed-Forward Neural Network

As shown in Figure 2, the feed-forward neural network is used to convert the categorical features that are sparse and high-dimensional into a low-dimensional and dense real-valued vector, which is referred as an embedding vector [1]. These vectors are randomly initialized, and during the training of model the final loss function is minimized by training the values [1]. Hidden layers of the neural network get as input these low-dimensional dense embedding vectors in the forward pass [1]. Specifically, the following computation is performed by each of the hidden layers:

=

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Where the activation function is represented by f and the layer number by [1]. is the activations, b is the

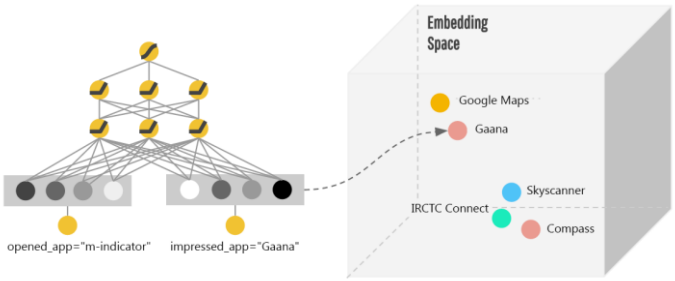

Figure 2: Deep Model. Adapted from " Wide & deep learning for Recommender systems", by H. Cheng et al, *arXiv preprint arXiv:1606.07792*. 2016

bias, and W is the model weights at the th layer [1].

As seen in Figure 1, the app recommended by the launcher just after opening "m-indicator" app was in the same *category = "travel."* A user may not be looking for another travel app here. By training the deep component for the android launcher, the recommendation engine learns lower-dimensional representations for each input query [1]. Thus, in the embedding space, generalization takes place by matching items that are close to each other [1]. In Figure 2, the model learns the feature *AND(opened_app = "m-indicator", impressed_app = "gaana").* A user checking for train timings in the m-indicator app may like to listen to songs during his/her transit, and, hence is recommended the app "gaana" by the launcher.

c. Training Wide & Deep Components

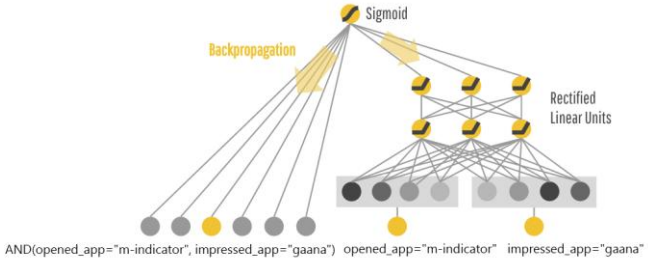

Figure 3: Wide & Deep Model. Adapted from " Wide & deep learning for Recommender systems", by H. Cheng et al, *arXiv preprint arXiv:1606.07792*. 2016

The two models are merged using a weighted sum of their o/p log odds as the prediction, and joint training is done by feeding the prediction to a common logistic loss [1]. During training, this model takes as input from the wide linear model, the feed-forward neural network and their weighted sum into consideration [1]. The wide linear model complements the limitations of the deep component by making use of cross-product feature transformations [1]. The gradients obtained from the output of this model are back-propagated to both the constituent models simultaneously using mini-batch stochastic optimization [1]. Illustration of the combined models is given in Figure 3.

The model's prediction for a logistic regression problem is:

$$P(Y = 1|x) = \sigma(w_{wide}^{T}[x, \phi(x)] + w_{deep}^{T} a^{(l_f)} + b)$$

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

Where the binary class label is represented by Y, the sigmoid function is represented by σ(.), the cross-product transformations of the original features x are represented by φ(x), and the bias for the model is represented by b. [1] The weights of the wide model are represented by the vector w_{wide} and the weights of the feed-forward neural network are represented by w_{deep} which are used on the final activations $a^{(l)}_f$ [1].

In Figure 3, the deep model is combined with a wide model to prevent over generalization of features. The sparse features like *opened_app = "m-indicator"* and *impressed_app = "gaana"* are present in both the models [1]. The prediction errors that surface when training the model are backpropagated to both models to train the parameters of the model [1]. The wide component memorizes the sparse rules using cross-feature transformations, while the deep feed-forward neural network uses embedding vectors to generalize similar itemsets [6].

2) Generalized Bayesian Hierarchical Linear Model

This model assumes that there are M number of users in the system. Providing recommendations that are relevant to a user is the main aim of the system [2]. The system builds a user profile from the user's history, for

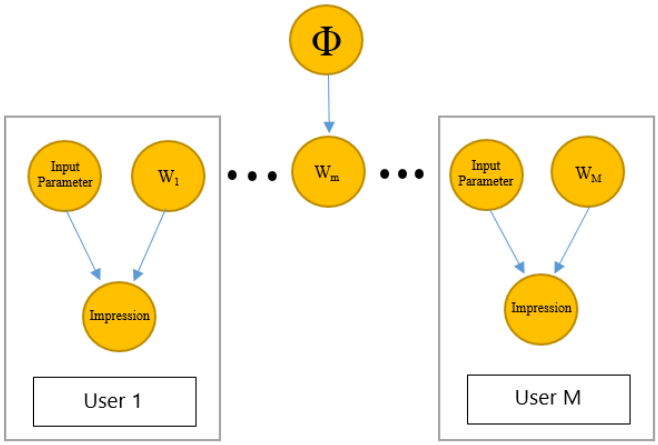

Figure 4: Bayesian Hierarchical Model. Adapted from "Efficient Bayesian hierarchical user modeling for recommendation system," by Y. Zhang and J. Koren, pp. 47–54, Jul. 2007

each user.

The Figure 4 shows M users where each user model is a K-dimensional random vector w_m, where m = [1, 2, …, M] is the index for each user [2]. The user models are taken as sample randomly from a distribution [2]. The system can predict the impression of an input parameter using a function given an estimation of w_m. [2] This model is called generalized Bayesian Hierarchical Linear model when is a wide linear model. [2] The limitation of the cold start problem is overcome by borrowing information from the user models present in the prior and then estimating reliably the user model wm for new users [2].

The population distribution is a assumed to be a Gaussian distribution governed by an unknown hyperparameter Φ. [2] The mean of this distribution is given by a K-dimensional vector $\mu = (\mu_1, \mu_2, \ldots, \mu_k)$ and the covariance matrix of the Gaussian is represented by [2].

The parameters represented by $\theta =$ are estimated and the joint probability for all the variables in the model, which includes the data and the system parameters, is:

$$P(D, \theta) = P(\phi)\Pi_m P(w_m|\phi)P(y_{m,j}|x_{m,j}, w_m)$$

IJARSE
ISSN (O) 2319 - 8354
ISSN (P) 2319 - 8346

For a user m, represents a dataset that is associated with that user and j = [1, 2, …, J_m] represents the index for that set [2]. This *m*th user's *j*th training itemset is a K-dimensional vector given by $x_{m,j}$ [2]. $y_{m,j}$ is a scalar representing the impression of input parameter $x_{m,j}$.

The output of EM algorithm is dependent on unobserved latent variables w and gives the results of parameter Φ [2].

IV. CONCLUSION

Recommendations based on user and contextual features can significantly enhance the user experience of Android users. Currently, the recommendation engine suffers from the cold start problem; wherein it takes a significant amount of data to build individual and independent user profiles that can be used to predict when and where a user makes use of a particular application, analyze the usage of the application to provide installation/uninstallation suggestion. The wide and deep learning models complement each other by using both memorization and generalization of features.

This paper proposes a recommender system for an Android launcher that makes use of wide & deep models to learn a user profile over a large amount of data and Bayesian linear hierarchical model to improve the initial performance for the users by borrowing data from existing user profiles. A Bayesian hierarchical model is used to generate a basic training set, that is used until enough data is collected to implement the wide and deep models.

REFERENCES

[1] H.-T. Cheng et al., "Title: Wide & deep learning for Recommender systems," 2016. [Online]. Available: http://arxiv.org/abs/1606.07792. Accessed: Sep. 2, 2016.

[2] Y. Zhang and J. Koren, "Efficient bayesian hierarchical user modeling for recommendation system," pp. 47–54, Jul. 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1277752. Accessed: Sep. 29, 2016

[3] Zigoris, Philip, and Yi Zhang. "Bayesian adaptive user profiling with explicit & implicit feedback." In Proceedings of the 15th ACM international conference on Information and knowledge management, pp. 397-404. ACM, 2006.

[4] Yu, Kai, Volker Tresp, and Shipeng Yu. "A nonparametric hierarchical bayesian framework for information filtering." In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 353-360. ACM, 2004.

[5] S. Rendle. Factorization machines with libFM. ACM Trans. Intell. Syst. Technol., 3(3):57:1–57:22, May 2012.

[6] H. Cheng, "Wide & Deep Learning: Better Together with TensorFlow," *Google Research Blog*, 2016.