Method for the Development of Recommendation Systems, Customizable to Domains, with Deep GRU Network

Arseny Korotaev[1] and Lyudmila Lyadova[2]
[1]Department of Computer Science, Perm State National Research University, Perm, Russia
[2]Department of Information Technologies in Business, National Research University Higher School of Economics, Perm, Russia

Keywords: Deep Learning, Knowledge-based Recommender System, Ontology, Customization, GRU.

Abstract: The GRU-based recurrent neural networks (RNN) for constructing recommendation systems are proposed. Such systems are mainly developed by large companies for specific domains. At the same time, small companies don't have the necessary resources to develop their own unique systems. Therefore, they need universal recommendation system (or recommender platform) automatically customized for a specific domain. This system allows to develop own recommendation system from scratch for companies whose services are under development. The RNN-based approach is proposed for session-based recommendation with automatically modelling of the domain. This approach is based on the content analysis of the web sites. Several modifications to classic RNNs such as a ranking loss function that make it more viable for this specific problem are considered. General scheme of the approach and architecture of the recommendation system based on proposed scheme are described in this paper.

1 INTRODUCTION

Currently, in the field of information and communication technologies, there is a need for systems that can guess the preferences and needs of users and offer suitable solutions. Such systems are named recommendation systems and have a lot of applications (Carlos, 2016).

Large companies such as Amazon, Apple, eBay, Yandex, etc., use recommendation systems as part of their products to offer relevant information to users. Recommendations are formed based on individual preferences of the user. This approach simplifies user work with a large amount of data, helps to reduce the time search for the desired solution, ensuring the receipt of relevant information.

Such systems are mainly developed by large companies to solve specific domain problems: Amazon developed its system for the sale of physical goods, Netflix for video content. At the same time, small companies don't have the necessary resources to develop their own unique systems. Therefore, there is a need for universal recommender platform that are automatically customized for a specific domain, this will help to solve the problem of the need to develop own recommendation system from scratch for companies whose services are under development.

The development of recommendation systems is based on the generally accepted methods of machine learning, so we can formulate the task of developing a universal solution based on these methods. The adjustment for the subject area can be performed by automatically modeling the subject area based on the content analysis of the web site.

A recommendation system adapted to a specific subject area should be based not only on the model of the recommendation system, its subject area and data collection tools, but also include data analysis tools that must ensure compliance with the requirements guaranteeing the quality of the results: relevance, uniqueness, completeness, structuredness, lack of redundancy, etc. Only with the availability of qualitative data is it possible to make decisions that most fully meet the needs of companies and users.

2 BACKGROUND

There are some software decisions for the development of recommendation systems:

- *Services with the module of recommendations for a certain data domain*. Such approach is

Korotaev, A. and Lyadova, L.
Method for the Development of Recommendation Systems, Customizable to Domains, with Deep GRU Network.
DOI: 10.5220/0006933302310236
In *Proceedings of the 10th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2018) - Volume 2: KEOD*, pages 231-236
ISBN: 978-989-758-330-8

Table 1: Comparing of the existing decisions for recommendation system development.

Evaluation criterion	Recommendation systems as a part of the completed services	Research prototype (Unresyst)	Libraries (Crab, LensKit, RankSys)	Prediction IO
Universality	0	2	1	2
Ability to integrate	0	0	0	1
Existence of components for data collection	2	0	0	0
Existence of the data analysis module	1	0	0	0
Accessibility	0	0	2	2

used, for example, in Apple, Netflix, Amazon and other companies.

- The *research projects* aimed at creation of the *universal platform* ensuring functioning with different data domains (Unresyst).
- Different *libraries* used by developers for creation of the own recommendation system.
- The *universal recommendation system* (Apache Prediction IO) as the *completed software product*.

Comparing of the existing decisions is given in table 1.

The following scale for estimation of decisions is used: 0 – the decision does not perform the specified function; 1 – the decision partially performs the specified function; 2 – the decision completely performs the specified function.

Comparing shows that *Prediction IO* most of all meets the delivered requirements: it can be used as the module which is built in the systems of third-party developers. However it is not integrated completely with the main service. Besides, there are no modules for the data storing and analysis which are necessary during creation of recommendation system (the completed services having the module of recommendations surely have also these means).

Thus, revealed at the existing solutions of restriction specify relevance of the universal recommendation system development.

Recommendation system is used for estimating users' preferences on items they have not seen (Duchi, 2011). There mainly exist three types of recommendation tasks based on the forms of outputs: rating prediction, ranking prediction (top-n recommendation) and classification.

Two main elements are involved in recommendations:

- *Recommender elements*. Elements can be characterized by complexity, value or utility. The value of an element can be positive, if the element is useful to the user, or negative, if the element does not fit it.
- *Users*. Users can have different goals and characteristics. The central role is played by the user model.

Transaction is a record of the interaction between the user and the system, a kind of logs necessary for the system to issue recommendations. For example, a transaction can contain a record of an item selected by the user and a context description. Additionally, it may contain a user feedback (rating of the selected item).

The *main classes of recommendation systems* based on different approaches described below.

The *system based on content* recommends items that are similar to those that the user liked in the past (Chen, 2017). The similarity of elements is calculated on the basis of features related to the characteristics of the compared elements. For example, if a user appreciates a movie that belongs to the genre of comedy, the system will recommend other films from this genre.

The easiest way to implement *collaborative filtering* approach is to recommend the user elements that other users with similar tastes liked in the past. The similarity of the taste of two users is calculated on the basis of the similarity in the history of the user rating. Collaborative filtering is considered to be the most popular and widely applied method in advisory systems (Deng, 2017; He, 2016).

The "*demographic*" system recommends items based on the user's demographic profile. It is assumed that various recommendations should be created for different demographic niches. Many websites adopt simple and effective solutions based on the personalization of demographics. For example, users are redirected to special departments of websites based on their language or country or the offer can be

customized according to the age of the user.

Knowledge-based systems recommend elements based on specific knowledge of the domain, about how some elements meet the needs and preferences of users and, ultimately, how the element is useful to the user. In these systems, the similarity function assesses whether the user's needs (problem description) correspond to the recommendation (task solution). Here the similarity can be directly interpreted as the utility of the recommendation for the user.

The "*social*" system recommends items based on the preferences of the user's friends. This method follows the rule "Tell me who your friends are, and I'll tell you who you are". Experience shows that people tend to rely more on recommendations from their friends than on recommendations from similar, but anonymous individuals (Carlos, 2016). This observation, combined with the growing popularity of social networks, generates a growing interest in systems of this kind. This type of system receives information about the social relationships of users and the preferences of the user's friends. The recommendation is based on a ranking that has been exhibited by the user's friends.

The *hybrid* system (Yi Zuo, 2016) is based on a combination of methods described above.

The above approaches have become widespread and have proved themselves well. However, there are more modern solutions. In this work we use recurrent neural networks of the GRU (Gated Recurrent Unit) type. Such networks are well suited for creating a model for predicting the next event along the chain of previous ones.

3 ARCHITECTURE OF THE RECOMMENDATION SYSTEM

The recommendation system consists of the following main components:

- Data collection and processing module.
- Module of data mining and recommendations generation.

This architecture should also provide an open API for integration with the user's service.

The main components of the recommendation system architecture are shown in Figure 1.

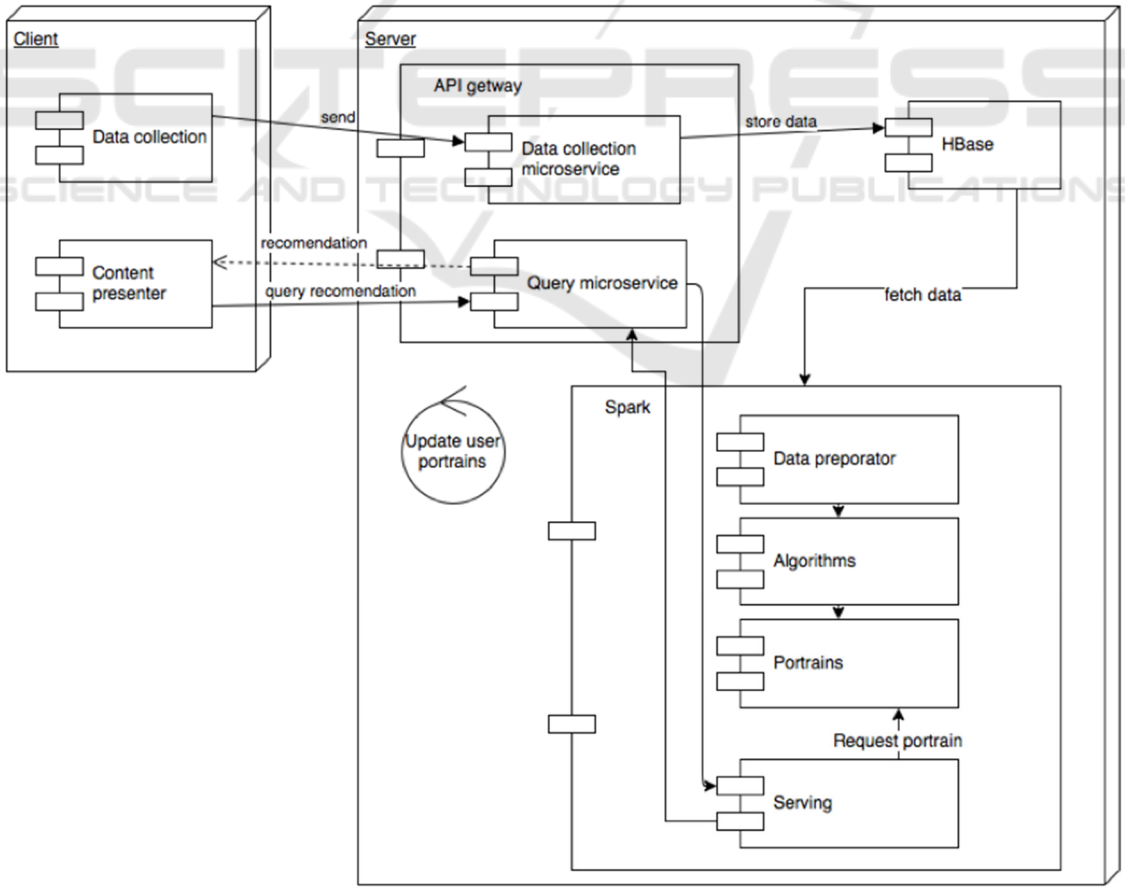

Figure 1: The recommendation system architecture: main components.

3.1 Data Collection and Processing Module

The requirements for the module are following:

- The module must have a programming interface (API) for receiving events from clients.
- The module must have an events store.
- The system must quickly process data having an exponential increase.

Let describe the above scheme in more detail.

The system has a *client-server architecture*.

Clients are "thin". Client must have at least two modules:

- *Data collection module*. Implements collection and sending of data about the user's behaviour on the server.
- *Content presentation module*. Individual for each client. The user ID asks the recommended content from the server.

Clients interact with the server using the REST protocol.

The *server* consists of API-part that implements a micro-service architecture and a data processor based on *Keras*.

There are two *API services*:

- *Data collection* receives data from clients and sends it to the non-relational database management system HBase.
- *Inquiry* accepts requests from customers for recommendations.

Modules functions are:

- *Data preparation*, *algorithm*: data preparation and model training takes place in this section. The data is selected from HBase and the Data Preparatory module step by step turns the data into a “working” format, generates the behaviour patterns given to the Algorithm module, produces system training and user model formation.
- *Portraits* store user models.
- *Serving module* accepts requests and gives recommendations.

3.2 Data Mining Module

Traditional neural networks have a significant limitation – they cannot remember information. For example, if the task is to classify what is happening in every frame of the film, then the classical neural network will not be able to use its previous conclusions for further solutions. Recurrent networks are aimed at correcting this shortcoming, since they contain cycles that allow to store information.

The same task is also in the recommendation systems. A certain history of interaction with the user accumulates. This information is important for constructing an accurate recommendation. The method based on recurrent networks is flexible and allows you to use any information about the objects of interest. In addition, importantly, the sequence of views is taken into account, which allows you to give preference to the latest events and, at the same time, adequately respond to a sharp change in mood. Another important plus is to get new recommendations user does not need to recalculate the model, he can get answers in real time.

There are two types of *recurrent networks* (Xie, 2016):

- LSTM (*Long short-term memory*). Long-term memory is a kind of recurrent neural networks capable of memorizing long-term dependencies. They were introduced by Sepp Hohrater and Jürgen Schmidhuber in 1997. LSTM networks were developed to address the problem of long-term dependencies. Remembering information for a long time is one of the main features of these networks, which does not require long-term training.
- GRU (*Gated Recurrent Unit*). Recurrent module with closures. Introduced by Cho, et al. (2014). This approach combines the "forget" valves and the input valves into a single update valve. The resulting model is simpler than the classical LSTM model.

If we compare models, then:

- GRU is less redundant, is trained 20-30% faster.
- The GRU almost always exceeds the LSTM in terms of work quality. At the same time it was shown that with a certain initialization of the LSTM cell we have almost no quality difference. But in fact, even with this initialization, the GRU is often better.

3.3 Ontology based Knowledge Representation Module

Ontologies are now regularly used in recommendation systems in combination with machine learning, statistical, user profiling and specific domain heuristics. Commercial recommendation systems usually either support simple product ontologies (e.g. books) that can then

be used with heuristics or have a large community of users actively evaluating content (for example, movies) suitable for collaborative filtering. More research-oriented systems of recommendations use a much wider range of methods that offer advantages such as improved accuracy combined with constraints such as feedback requirement or intrusive monitoring of user behavior over long periods of time (Middleton, 2009).

Ontology is a powerful tool for profiling users. Ontology can contain all sorts of useful knowledge about users and their interests, for example, related to scientific subjects, technologies underlying each subject area, projects over which people work, etc. This knowledge can be used to bring out more interests than it can be seen simply by observation.

We try to use inference to improve user profiles. Is-relationships in the thematic ontology are used to evoke interest in more general topics of the superclass. This conclusion has the effect of rounding profiles, making them more inclusive and tuning them to the broad interests of the user.

We try to apply time decay to the observed behavior events for the formation of the main profile. Then the output is used to enhance the profile of interests, with the 50% output rule applied to all ontological relationships, up to the root class, for each observed event.

The events of interest were chosen to balance the feedback in favor of explicitly provided feedback, which is likely to be the most reliable. The value of the 50% output was chosen to reflect a decrease in confidence that you are deviating from the observed behavior that you are moving. Determining the optimal values for these parameters will require further empirical evaluation.

Recommendation systems suffer from a cold start problem (Middleton, 2009), where the lack of initial behavioral information significantly reduces the accuracy of user profiles and therefore recommendations. This low performance can keep users from accepting the system, which, of course, does not allow the system to receive more data about behavior; it is possible that the recommendation system will never be used enough to overcome its cold start.

We try to apply external ontology containing domain specific data. The knowledge stored in the external ontology is used to initial load user profiles in order to reduce the effect of cold start. External ontology providing a solid basis for communication. Knowledge stored in an external ontology is used to determine historical interests for new users, and a network analysis of ontological relationships is used to identify similar users whose own interests can be used to load a new user profile.

3.4 Recommendations with GRU

RNN have been developed to model sequence data. The main difference between RNNs and ordinary feed forward deep model is the existence of internal state unit. Standard RNNs update hidden state with function

$$h_t = g(Wx_t - Uh_{t-1}),$$

where g is logistic sigmoid function of x_t. An RNN outputs a probability distribution over the next element of the sequence, given its current state h_t.

A Gated Recurrent Unit (GRU) is more advanced model of RNN that aims to solve vanishing gradient problem. GRU gates essentially learn when and by how much to update the hidden state of the unit (Razvan, 2012). The activation of the GRU is a linear interpolation between the previous activation and the candidate activation $\hat{h}_t$:

$$h_t = (1 - z_t)h_{t-1} + z_t \hat{h}_t,$$

where the update gate is given by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}),$$

while the candidate activation function $\hat{h}_t$ is computed in a similar manner:

$$\hat{h}_t = \tanh(Wx_t + U(r_t \odot h_{t-1})),$$

and finally the reset gate r_t is given by:

$$r_{t=}\sigma(W_r x_t + U_z h_{t-1}).$$

We used the GRU-based RNN model for recommendations. The input of the network is the current *session* while the output is the item of the next event in the session. The state of the session can either be the item of the actual event or the events in the session so far. In the former case 1-*of*-*N* encoding is used, i.e. the input vector's length equals to the number of items and only the coordinate corresponding to the active item is one, the others are zeros. The latter setting uses a weighted sum of these representations, in which events are discounted if they have occurred earlier. For the stake of stability, the input vector is then normalized. We expect this to help because it reinforces the memory effect: the reinforcement of very local ordering constraints which are not well captured by the longer memory of RNN. We also experimented with adding an additional embedding layer, but the 1-*of*-*N* encoding always performed better.

The core of the network is the GRU layer(s) and additional feedforward layers can be added between the last layer and the output. The output is the

predicted preference of the items, i.e. the likelihood of being the next in the session for each item. When multiple GRU layers are used, the hidden state of the previous layer is the input of the next one. The input can also be optionally connected to GRU layers deeper in the network, this improves performance. See the whole architecture on Figure 2, which depicts the representation of a single event within a time series of events. The final sentence of a caption must end with a period.

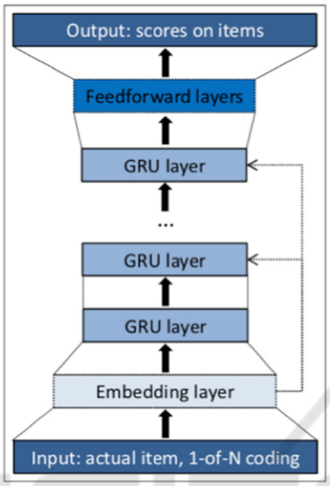

Figure 2: General architecture of the network.

Since recommendation systems are not the primary application area of recurrent neural networks, we modified the base network to better suit the task. We also considered practical points so that our solution could be possibly applied in a production environment.

4 CONCLUSION

Theoretical significance of the work is to develop universal approach to develop recommendation systems based on data from different domains.

To improve the quality of recommendations we suggested to use the composition of data analysis methods, which can efficiently solve the tasks of clustering, searching for duplicates and associations in the recommender systems of several regions spectra.

This approach can be used to develop recommender platform for groups of domains with specific requirements, which are difficult to take into account when developing.

The versatility of the developed system provides its advantages over other products on the market: the service can be configured to needs of many companies operating in various fields, while do not need the cost of its development. Installation (or connection with existing service) and tuning can be carried out in the shortest possible time and does not require a high qualification of company personnel.

REFERENCES

Carlos, A Gomez-Uribe, Hunt, N., 2016. The recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems* (TMIS). 6, 4 (2016), 13.

Chen, J., Zhang, H., He, X., Nie, L., Liu, W., Chua, T., 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In: *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*.

Deng, Sh., Huang, L., Xu, G., Wu, X., Wu Zh., 2017. On deep learning for trust-aware recommendations in social networks. *IEEE transactions on neural networks and learning systems*. 28, 5.

Duchi, J., Hazan, E., Singer, Y., 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

He, R., McAuley, J., 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of the 25th International Conference on World Wide Web*.

Middleton, S. E., Roure, D. D., Shadbolt, N. R., 2009. Ontology-Based Recommender Systems. *In: Staab S., Studer R. (eds) Handbook on Ontologies. International Handbooks on Information Systems*. Springer, Berlin, Heidelberg

Razvan, P., Tomas, M., Yoshua, B., 2012. *On the difficulty of training recurrent neural networks*. arXiv preprint arXiv:1211.5063.

Xie, R., Liu, Z., Yan, R., Sun, M., 2016. Neural Emoji Recommendation in Dialogue Systems. *arXiv preprint arXiv*:1612.04609.

Zuo, Yi., Zeng, J., Gong, M., Jiao, L., 2016. Tag-aware recommender systems based on deep neural networks. *Neurocomputing* 204 (2016).