A Loan Application Fraud Detection Method Based on Knowledge Graph and Neural Network

Qing Zhan, Hang Yin

Shanghai Dianrong Information Technology Co., Ltd.

{qing.zhan, hang.yin}@dianrong.com

ABSTRACT

Dianrong, a tech-driven internet finance company, provides loans to a large number of people and small business users. The ability to predict fraud from loan applications is key to the company's business. Based on published literature on fraud detection techniques, features have to be extracted manually for further rule design or machine learning. But as fraudulent behaviors change over time to avoid detection, simple features or rules become obsolete quickly. Normally, we have to extract hundreds of features, which is a time and resource consuming process. This paper proposes a new way to extract features automatically from a borrower's phone network graph using neural networks, which not only overcomes the above issue, but also captures features that are hard to fake. This method has yielded strong results in reality.

CCS Concepts

• **Applied computing → Electronic commerce → Online banking**

Keywords

Loan Application Fraud, Knowledge Graph, Neural Network.

1. INTRODUCTION

A type of risk in finance is called Application Fraud Risk. In the case of loans, fraud risk is when a group or individual does not intend to repay the loan from the moment the application is submitted. The fraudster often provides fake identity, contacts, equipment, asset information, etc.

Banks have been fighting with it for a long time. They build a rule system based on expert experience (hereinafter referred to as an "expert rule system"). Each time a fraud event is encountered, the system records the event's characteristics to form a rule. Next time an event with similar characteristics is encountered, the behavior, rule system will either trigger a warning or block the application. However, it is difficult to capture all fraud characteristics. A specific rule can be created only when a fraud event occurs. In other words, the expert rule system can't predict what kind of fraud will happen next time.

Recently, consumer finance loans become popular. They are small, scattered and highly concurrent. A traditional expert rule system alone cannot effectively detect fraud for these scenarios.

ICIAI 2018, March 9–12, 2018, Shanghai, China

ACM ISBN 978-1-4503-6345-7/18/03…$15.00

DOI: https://doi.org/10.1145/3194206.3194208

We combine knowledge graph and machine learning, which have been making big progress recently, and apply them to consumer finance fraud detection. The general algorithm is:

1)Construct a graph based on related data (We currently focus on the borrower's phone data);

2)Collect neighbor information for all vertices as context for these vertices, based on Node2Vec [1] with extension by adding vertex property to random walk path;

3)Learn a Word2Vec model with the context;

4)Predict the new borrower's probability of default based on the similarity of their phone data with historical data as returned by the model.

Our major contributions are:

1)Identify a data source that is difficult for fraudsters to fake: borrower call history;

2)Automatically extract features;

3)Extend Node2Vec by adding vertex property in path.

2. RELATED WORK

The field of loan application fraud detection is making progress in 2 directions:

1)Use machine learning to learn patterns which are hard to observe for humans, to complement the expert rule system;

2)Use graph-based data, such as social networks, to design rules. For example: A high risk rule can be "borrower A's father and borrower B's wife use the same phone number".

Direction 1: Researchers have paid much attention to credit scoring in application fraud detection. Credit scoring is defined as a statistical method used to predict the creditworthiness of a customer [2]. Its models are usually built using linear methods, such as logistic regression [4,5]. Other artificial intelligence and machine learning techniques that are used include Bayesian, decision trees, support vector machines, ensemble methods and neural networks [3,6,7,8,9,10]. In financial institutions, logistic regression and decision tree ensemble methods, such as Gradient Boosting Decision Tree (GBDT), are the most frequently used models. They heavily rely on handcrafted feature engineering.

Direction 2: Use graph information, mainly social network data, to extract features (such as Degree Centrality, Connected Components, Triangle Count). Then, design rules, such as the following:

1)A borrower connected to many other fraudulent borrowers is likely to be fraudulent;

2)If borrower A's contact person of happens to use the same phone number as borrower B's contact person, borrower A is likely to be a fraudster. The phone number is likely to be a fraud agent number;

3)Unearth the contact network of borrowers with lost contact information.

Recently, graph embedding is popular in both academia and industry, because it can take advantages from both knowledge graph and machine learning. Each vertex in a graph is embedded into a low dimensional vector space. Embedding preserves the structural similarity or proximity among the vertices in the original graph. Most implementations learn vertex representations by using random walk [1,12] to uncover graph structure. These methods have many real-world applications [16], such as Recommendations [11], Social Network Analysis [12,17], Natural Language Processing [13] and Knowledge Bases [14,15], but are rarely seen in loan application fraud detection.

3. SYSTEM ARCHITECTURE

The whole process is divided into 2 parts: training process and online process. The architecture is shown in Figure 1.

The training process is a scheduled job, running on a Spark cluster. Its output, random walk path file and Word2Vec model, is saved to Hadoop file system(HDFS). The steps are:

1)Extract borrower call data from loan transaction data warehouse;

2)Construct a Spark Graphx diagram;

3)Do a random walk based on an enhanced Node2Vec algorithm and save paths to a file;

4)Learn a Word2Vec model from the paths and save it to HDFS.

The online process is a scalable web service, providing Representational State Transfer (REST) API for fraud detection. The steps are:

1)Load random walk path from HDFS;

2)Do a simulated walk to get walk path;

3)Do incremental learning with the model learned from the training phrase;

4)Predict likelihood of default for the borrower.

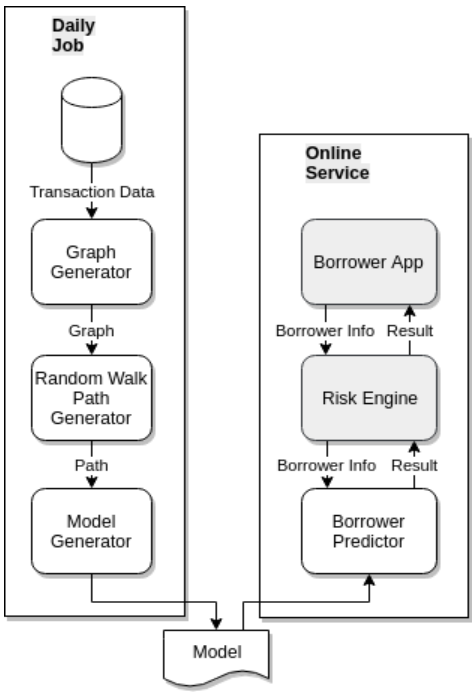

Figure 1. System architecture

4. FRAUD DETECTION DETAIL

Details are divided into 5 sections: 4.1 describes how to pick phones from the borrower's call history to construct a call network graph; 4.2 describes how to choose parameters to do a random walk; 4.3 describes the Word2Vec library choice based on our online learning requirements; 4.4 describes the optimization and trade-off made for real-time requirements; 4.5 gives the result of similar phone numbers returned by Word2Vec.

4.1. Graph Construction

We get the borrower's call history data from a third-party provider, combining it with our loan transaction data to build a call network graph. All vertices are phones (of either the borrower or their contact), with calls made as connections. As the total number of phone contacts is very large, it is impractical to put all of them into the graph. Thus, we build filters: 1) remove well-known public service numbers, such as telecom providers; 2) set a number $N_{contact}$, to only keep phone numbers with $N_{contact}$ or more borrowers as contacts.

A sliding window is used to limit the graph size. In our case, we have around 1M vertices and 10M edges.

4.2. Random Walk

In general, there exist 3 different types of similarity between vertices: structural similarity, semantic similarity and proximity. Structural similarity pays attention to the vertex's role in the graph, such as hub or outlier; semantic similarity pays attention to the attributes of vertices and edges; proximity pays attention to the distance between vertices. We will focus on semantic similarity and proximity in our research.

Node2Vec is chosen for random walk. By observing the graph, the following is found to be a rare scenario: walk from vertex t to vertex v. There exists vertex x_1 connected to both t and v, as shown in Figure 2. So we only need to consider 2 actions for the next step: go further or go back.

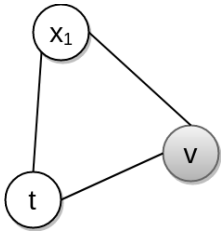

Figure 2. Rare connection scenari

Eventually, we would like the recommended vertices to be similar in proximity. The underlying rationale is that fraudsters tend to borrow money from multiple financial institutions. It's highly possible that they are connected by some shared fraud agency or debt collection phone numbers. For this reason, we prefer to take a small step further instead of going back. The result is like a star, shown in Figure 3, in which the gray vertex is the start point and white vertices are those in the walking path. We may encounter a borrower phone or non-borrower's phone during the walk. A non-borrower's phone won't be put in the path for the Word2Vec model learning later (model path). So, the length of model path is roughly half of the actual walking path. Based on the above discussion, we get the following Node2Vec parameters. Walk parameters are in black.

dim=40, ***wl=6, numWalks=30,*** *windowSize=5,* ***p=10.0, q=0.1***

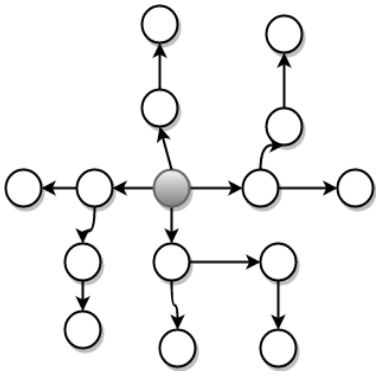

Figure 3. Random walk path

The original Node2Vec path only contains vertex ID, meaning only graph structure is taken into account. Actually, vertex attributes, such as third-party credit score, sex, etc, can all be important. We will combine attributes to a long value, as Figure 4 shows. If the raw attribute has many values such as credit score, it will be divided into several levels. The combined attribute is put into the model path along with vertex ID, for example: (phone1, phone1_attr, phone2, phone2_attr...).

Figure 4. Attributes organization.

4.3. Word2Vec Model Creation

A random walk is used to learn a Word2Vec model. We choose Gensim Word2Vec lib because it supports incremental learning: new paths with new words that are not in the original model's vocabulary can be incrementally learned. This takes a short time if the new input is not big, which fits our online learning requirement.

4.4. Real-Time Optimization

The training process – generate a graph, complete a random walk, learn the whole Word2Vec model, then predict – takes a lot of time. This is unrealistic in the online scenario, which requires a response in under 5 seconds.

We use an incremental approximation approach to overcome this issue. First, save the random walk path during the training phrase. Then, during test or online implementation, load it to a map with phone as key and path as value. Finally, for each of the new borrower's contact phones, search the path from the map and append the new phone to the head of the path if found. If no path is found for the borrower, we would say we found no reason to reject the application. Take Figure 5 as an example, new incoming borrower_phone has contact phones 1 to N. There are 2 existing paths for contact_phone1: (phone1, phone2, phone3), (phone4, phone3, phone5). Here, contact_phone1 does not belong to a borrower, hence it doesn't appear in the paths. Then 2 new paths are created for the borrower_phone: (borrower_phone, phone1, phone2, phone3), (borrower_phone, phone4, phone3, phone5). The same is applied to the rest of the contact phones.

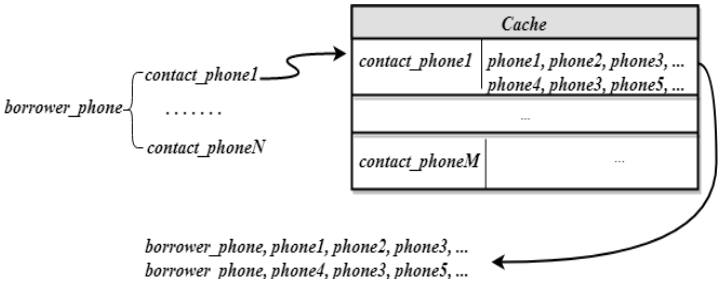

Figure 5. Random walk simulation

All the concatenated paths of one borrower's phone are sent to do an incremental Word2Vec learning. Then we can get the N most similar borrower's phones to calculate the new phone's default weight ($W_{overdue}$) with the following formula:

$$W_{overdue} = \frac{N_{bad}}{N_{all}} * AVG_{overdue}$$

Where N_{bad} is the count of bad borrowers (defined as if the borrower is late on a loan more than a threshold number of days, then he/she is bad), N_{all} is the count of all borrowers, $AVG_{overdue}$ is the average number of days the loan is late. This implementation keeps our API response time at around 1 second.

4.5. Word2Vec Recommendation Result

Figure 6 shows the result returned by Word2Vec for similar phone numbers for 98% of the cases. The white vertex is the one to be predicted, gray ones are its contact phones, black ones are contact's contact, hence the most similar phones returned by Word2Vec.

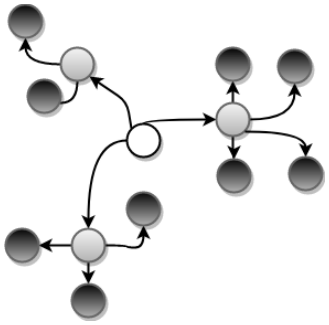

Figure 6. Most similar phones result.

5. EXPERIMENTS

Experiments are done from different perspectives to show the effect of phone networks on loan application fraud detection.

5.1. Used Alone for Detection

First, we use the model alone to detect fraud. The results are plotted as a Precision Acceleration-Recall curve (acceleration of the GBDT model Dianrong is using). We use this instead of a Precision-Recall curve because the latter touches on our business secrets. The result is shown in Figure 7. With the new model, we can lower the default rate by an additional 0.5% to 1%.

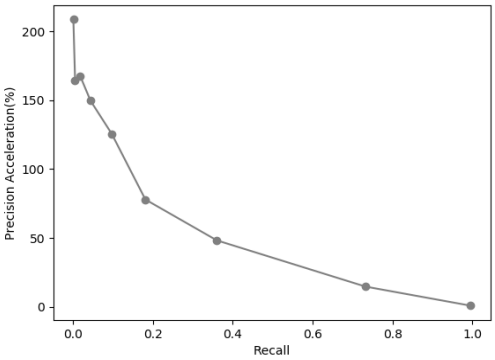

Figure 7. Acceleration to GBDT model.

5.2. Combined with Other Models

5.2.1. Default Weight as a Feature

The default weight mentioned in section 4.4 is used as a feature for a K Nearest Neighbor (KNN) model along with other features like third party credit score, sex, income etc. to do classification. Figure 8 shows feature importance. Features like t1, t2 are secret to the company. "weighted" is the default weight. The line add_KG_AUC in Figure 9 shows 0.02 improvement on Area Under Curve (AUC) compared to the original one.

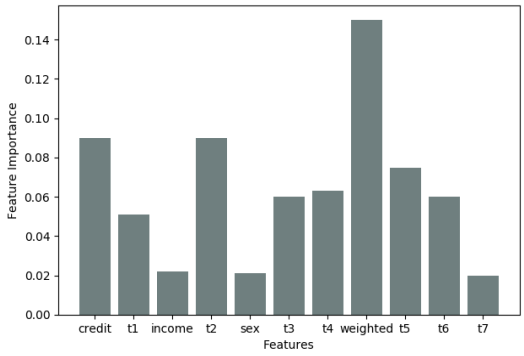

Figure 8. Feature Importance

5.2.2. Word2Vec Vector as a Feature

We also use the Word2Vec output vector as a feature to do the above test. The line add_word2vec_AUC in Figure 9 shows 0.05 improvement compared to original AUC.

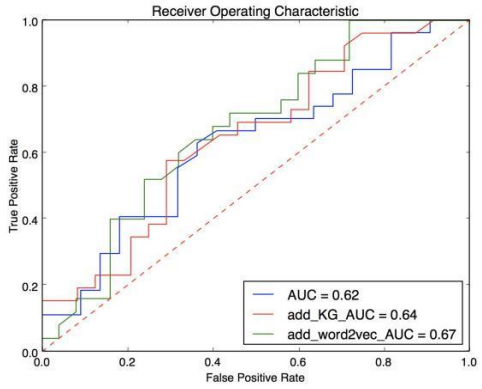

Figure 9. Improvement on AUC.

6. DISCUSSION AND CONCLUSION

We've proposed a new way to do loan application fraud detection by using borrower call history data. It has the following advantages: 1) it is hard to be faked by fraudsters; 2) features can be extracted automatically. We have used it in practice and seen improvements on original models. The bottleneck of the model is the noise in call history (too many irrelevant phone numbers).

We noticed the borrower's call history data are noisy, which may affect the prediction precision. It's planned that only the latest 1 month data are kept for analysis. We are also working on algorithms to identify shared fraud agency and debt collection phone numbers. With that, we can filter out more noise in the graph, and increase the precision eventually.

7. REFERENCES

[1] Grover, A., and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In International Conference on Knowledge Discovery and Data Mining. ACM.

[2] A. Ghatge, P. Halkarnikar. 2013. "Ensemble neural network strategy for predicting credit default evaluation," Int. J. Eng. Innov. Tech., vol. 2, pp. 223-225

[3] E. David. 2012. Bayesian inference-the future of online fraud protection. Computer Fraud & Security, 8-11.

[4] Bhattacharyya, S., Jha, S., Tharakunnel, K., Westland, J.C. 2011. Data mining for credit card fraud: A comparative study. Decision Support Systems 50, 602–613

[5] Dechow, P., Ge, W., Larson, C., Sloan, R. 2011 Predicting material accounting misstatements. Contemporary Accounting Research 28, 17–82

[6] M. Zareapoor, P. Shamsolmoali. 2015. Application of Credit Card Fraud Detection: Based no Bagging Ensemble Classifier. International Conference on Intelligent Computing, Communication & Convergence.

[7] A. Khashman, "Credit risk evaluation using neural networks: Emotional versus conventional models," Appl. Soft Comput., vol.11(8), 5477-5484, 2011.

[8] H. Ince, and B. Aktan, "A comparison of data mining techniques for credit scoring in banking: A managerial perspective," J.Bus. Econ..Manage., vol. 10(3), pp. 233-240, 2009.

[9] D. West, "Neural network credit scoring models," Comput. Oper. R., vol. 27(11–12), pp. 1131-1152, 2000.

[10] C. F. Tsai, and J. W. Wu, "Using neural network ensembles for bankruptcy prediction and credit scoring," Exp. Sys. Appl., vol. 34(4), pp. 2639-2649, 2008.

[11] Barkan, O., and Koenigstein, N. 2016. Item2vec: Neural item embedding for collaborative filtering. arXiv preprint arXiv:1603.04259.

[12] Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 701–710. ACM.

[13] Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. Line: Large-scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, 1067–1077. ACM.

[14] Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In AAAI, 2181–2187.

[15] Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In AAAI, 1112–1119. Citeseer.

[16] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, “Scalable graph embedding for asymmetric proximity,” in AAAI, 2017, pp. 2942– 2948.

[17] L. Liu, W. K. Cheung, X. Li, and L. Liao, “Aligning users across social networks using network embedding,” in IJCAI, 2016, pp.1774–1780