LINK PREDICTION IN CITATION NETWORKS

INF582 Text Mining and NLP Data Challenge

17 mars 2018

Team: Diss
Members:
DAI Yayun, SUN Wang, XIA Shiwen

1 Introduction

This project aims to predict missing links in a citation network of scientific papers. Two kinds of information are provided : some known links in the citation network ; meta and textual information of those scientific papers. In this case, we have two strategies to choose from : a graph-based algorithm uses only the known links to predict other potential links based on a degree of closeness between nodes ; a feature-based algorithm uses meta, textual and graphical features to build a binary classification model.

2 Feature engineering

To well describe similarities between two papers, we captured several features as listed below :

meta features[1]

- *year_diff* : The paper published before cannot cite those published after.
- *comm_authors* : Authors are largely probable to cite their own papers.

textual features

- *journal_sim* : The papers published in the same journal or the similar journal are more probable to be in the same domain and then to have links.
- *title_overlapping* : Title contains keywords which reveal the subject. Papers of the same subject are possible to have links.
- *abstract_sim* : Abstract describes the main idea of paper and indicates the subject or domain. Papers in the same domain are possible to have links. We tried several different document representations like **TFIDF** , **Google Word2Vec embedding**, **Gaussian distribution**[2] and **Word Mover's Distance** to calculate text similarities.

graphical features[3, 4, 5]

- *cn(common neighbors)* : more common neighbors, more liked to be linked.
- *aai(Adamic Adar Index)* : AAI of u and v is defined as $\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{\log|\Gamma(w)|}$, where $\Gamma(u)$ denotes the set of neighbors of u.
- *rai(Ressource Allocation Index)* : RAI of u and v is defined as $\sum_{w\in\Gamma(u)\cap\Gamma(v)} \frac{1}{|\Gamma(w)|}$.
- *pa(Preferential Attachment)* : PA score of u and v is defined as $|\Gamma(u)||\Gamma(v)|$.
- *jc(Jaccard coefficient)* : JC of nodes u and v is defined as $\frac{|\Gamma(u)\cap\Gamma(v)|}{|\Gamma(u)\cup\Gamma(v)|}$
- *rwr(Random Walk with Restart)* : RWR of nodes u and v is defined as $s_{uv}^{RWR} = q_{uv} + q_{vu}$ where $\vec{q}_u = (1-c)(I - cP^T)^{-1}\vec{e}_u$, $\vec{e}$ is the unit vector and $P_{uv} = 1/deg(u)$, c is a scalar parameter representing the probability of moving forward in random walk.

— *In-link difference* : the difference in the number of in-links of source paper and target paper, which evaluates the attracting force between two nodes.
— *tc(Target citation)* : the number of in-links of target paper. In the preferential attachment model, the probability of a new edge involving node x is proportional to the number of links of x. With this model, the node with more citations will receive more citations.

During our experiments, *Preferential Attachment* seems to have a negative influence on the prediction performance in almost all models that we built. It is thus eliminated quickly. Some features need massive matrix computation such as *Random Walk with Restart*, due to lack of computation and storage power, we abandon them despite their excellent performance in some reported works. We also try *Polynomial Features* with *scikit-learn*, but no significant improvement is achieved except the increase of computational complexity.

We note that some features are highly correlated with each other, especially the graphical features and the textual similarity features. Nevertheless, we keep them all and leave the machine learning models to adapt their weights automatically during the training process. Even though using more features doesn't necessarily improve the predictive ability of our models according to the feature importance analysis, the absence of some features like *target citation* and *year difference* will surely decrease the prediction performance. In case that some important features are wrongly ignored, we keep all these listed features except those with obvious side effects such as *Preferential Attachment* for our following model training.

3 Model tuning and comparison

Generally, we find that the performance of each model has relatively coherent dependency on features in our trials. A good choice of features will lead to a good performance for almost all models. Furthermore, for Logistic Regression, SVM and Neural Networks, a scaling of data is necessary before the training step. *StandardScaler* seems to perform better than *MinMaxScaler* according to our experiments. A 5-fold cross validation is performed for maximizing F-score in our parameter tuning step.

3.1 Logistic Regression

L2 penalty is applied to avoid overfitting. The inverse of regularization strength C is the parameter we mainly tune. The smaller C is, the less complex the model is. Even though the improvement is not very obvious, we can still find an optimal magnitude. In addition, we could set a tolerance of loss and a maximum number of iterations to control the training degree. Smaller tolerance and more iterations mean that the model fits better on training dataset, but may also introduce a risk of overfitting. From the Figure 1, we can see that the best C is 10^2.

3.2 SVM

We choose *LinearSVC* model from *scikit-learn*. As above, we still use L2 norm as penalization term. The parameter tuning focuses on the penalty parameter C. A large C may result in insufficient regularization, so may raise the risk of overfitting.

3.3 Random Forest

Random Forest is an ensemble method with bagging technique. As we do not have many features, we limit the *max_depth* of Decision Trees to less than 4. We choose mainly *n_estimators*, *max_depth* and *min_samples_split* to tune. More trees, deeper tree and fewer samples in leaves will better fit and give better performance.

3.4 AdaBoost & XGBoost

Boosting is another method of ensemble learning. We try AdaBoost and XGBoost in our project. Their basic estimator is Decision Tree classifier. *max_depth* of the decision tree and the *number of estimators* are two parameters we focus on. In practice, we limit the *max_depth* under 3. Figure 3 shows that 300 estimators is the best choice. For XGBoost, three *sample* parameters guarantee the randomness of instances used for each decision tree to thus work well on overfitting problem. A part of the dataset is used as validation set for early stopping. We choose *logloss* as the evaluation metric.

3.5 Neural Networks

The structure of networks and the penalty parameter are two main parameters to tune. We try different architectures, but the choose of architecture is, to some extent, intuitive. Generally, too many layers result easily in overfitting. We try several architectures with 2 hidden layers and 3 hidden layers. The results show that a network with 3 hidden layers of 30, 15, 5 neurons works the best. The choice of penalty parameter is based on tests of different values5. We think that a neural network with more carefully tuned parameters may perform even better.

Neural network model outperforms (Table 1) the previous models in our experiments. F-scores of 0.97457 and 0.97580 on public and private leaderboards are our best results of this project.

3.6 Ensemble

Finally, we try to combine the best results obtained via different models together and take a majority vote as the final prediction result. However, the performance of the combined result doesn't surpass that obtained by the neural network model.

4 Conclusion

The feature engineering is the most important factor of model performance. A good choice of features can improve significantly the accuracy of prediction. Parameter tuning is also important to avoid overfitting, and to guarantee a similar performance on local validation and on testing datasets.

The Neural network model outperforms all the other models in our project. Ensemble methods with bagging or boosting perform usually better than simple methods.

Appendices

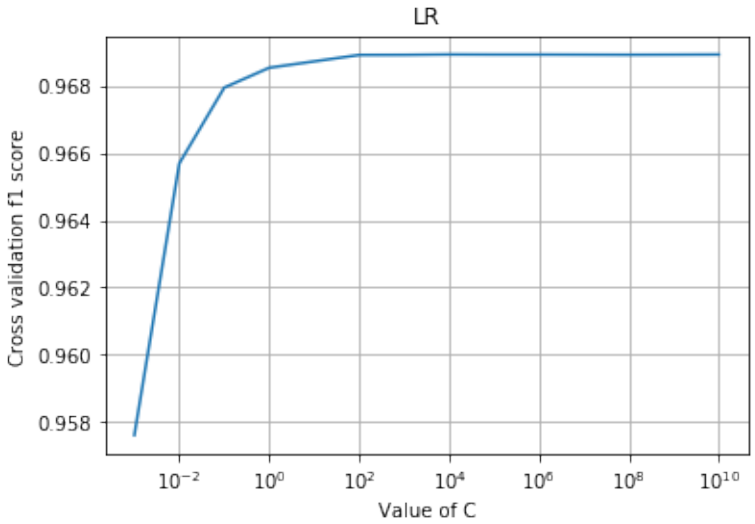

Figure 1 – Tuning penalty parameter of logistic regression

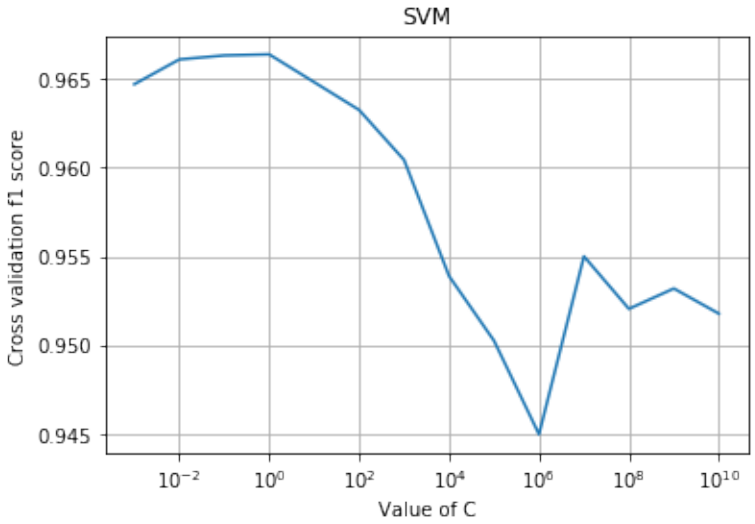

Figure 2 – Tuning penalty parameter of SVM

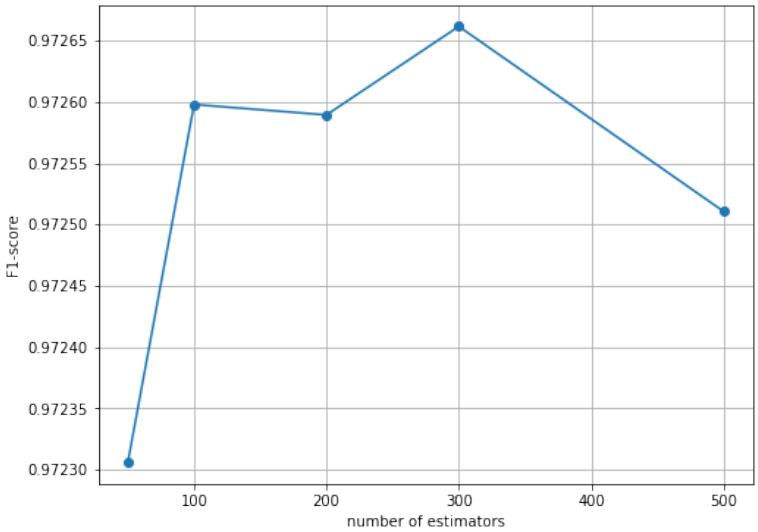

Figure 3 – Tuning number of estimators of AdaBoost

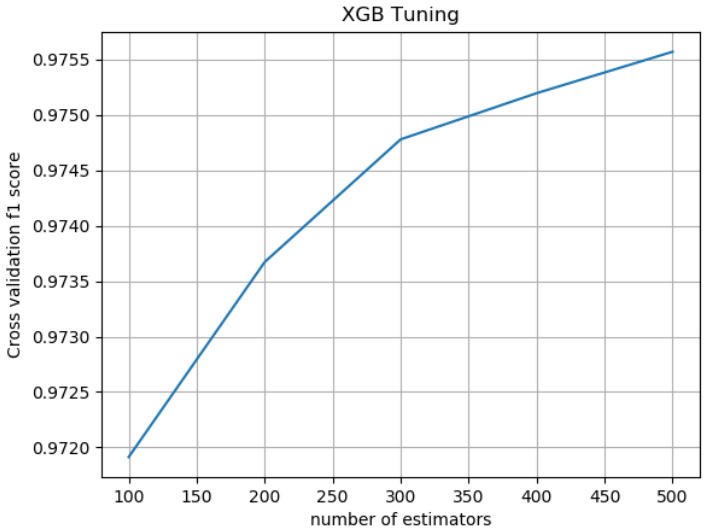

Figure 4 – Tuning number of estimators of XGBoost

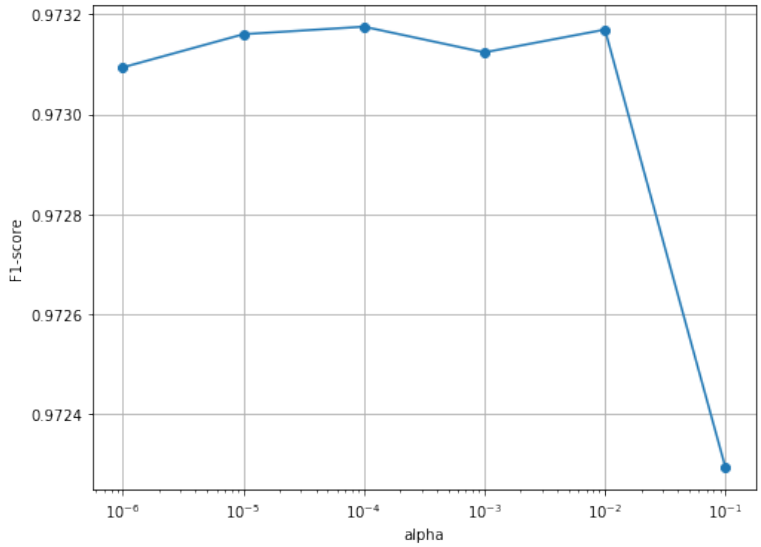

Figure 5 – Tuning penalty parameter of neural network

Table 1 – Performances for different models

Model	Local best	Best public	Best private
Logistic Regression	0.96894	0.96599	0.96661
SVM	0.96720	0.96416	0.96521
Random Forest	0.97408	0.96936	0.96836
AdaBoost	0.97266	0.97190	0.97120
XGBoost	0.97495	0.97345	0.97383
Neural Networks		**0.97457**	**0.97580**
Graph with JC*		0.95729	0.95732
Graph with RAI*		0.96005	0.95946
Graph with AAI*		0.95913	0.95885

* These results are obtained from graph-based algorithms, which considering just the graph structure, but not the textual information and other features of the nodes.
** Some local validation results are missing because we didn't note them down during the experiments.

Références

[1] Naoki Shibata, Yuya Kajikawa and Ichiro Sakata. Link prediction in citation networks. *Journal of the American Society for Information Science and Technology, 63(1), 78–85.*

[2] Giannis Nikolentzos, Polykarpos Meladianos, Franc¸ois Rousseau and Michalis Vazirgiannis. Multivariate Gaussian Document Representation from Word Embeddings for Text Categorization, *Proceedings of the 15th Conference of the European Chapter of the Association for computational Linguisticss : Volume 2, pp. 450–455.*

[3] L. Lü, T. Zhou. Link prediction in complex networks : a survey. *Phys A, 390 (2011), pp. 1150-1170.*

[4] D. Liben-Nowell, J. Kleinberg. The link prediction problem for social networks. *CIKM '03 : Proceedings of the Twelfth International Conference on Information and Knowledge Management, ACM, New York, NY, USA (2003), pp. 556-559*

[5] Alexis Papadimitriou, Panagiotis Symeonidis, Yannis Manolopoulos. Fast and accurate link prediction in social networking systems. *The Journal of Systems and Software, 85 (2012), pp. 2119– 2132*