A Personalized Company Recommender System for Job Seekers

Yixin Cai, Ruixi Lin, Yue Kang

Abstract

Our team intends to develop a recommendation system for job seekers based on the information of current employees in big companies. Several models are implemented to achieve over 60% success rate in classifying employees, and we use these models to help job seekers identify their best fitting company.

1. Introduction

Job seekers want to find the most satisfactory jobs through improving their resumes and working experience; however, they lack effective guidance for improvement. They may also be curious about the variant hiring criteria for different companies, and whether they possess certain qualities hiring managers are looking for. By exploring and analyzing employees' features of big companies, such as Apple, Facebook, and Google, our team proposes an effective recommendation system to match job seekers to their most suited companies.

Our recommendation is composed of evaluating the features of current employees and classifying them based on their companies. Our target companies include Apple, Facebook and Google. The employees' data are mainly retrieved from LinkedIn website. The procedure works as follows: 1) Collect employee data and extract features; 2) Train different multinomial classification models, eliminate poorly uncorrelated features, and keep features with high correlation; 3) Choose the optimal classification model; 4) For a job seeker, predict the best-fit company based on features of that person.

2. Related Works

Although there are few papers on company recommendation system, we could still find many research results on similar topics like personalized music, movie, products, webpages recommender systems, and group recommendations for social networks. Douglas Eck, Thierry Bertin-Mahieux and Paul Lamere.[1] developed music auto-tagging system using meta-learning algorithm. Claudio Biancalana[2] proposed a method to recommend movie applying neural networks. Yukun Cao and Yunfeng Li[3] designed a fuzzy-based system for consumer electronics to retrieve optimal products. Lihong Li, Wei Chu, John Langford and Robert E. Schapire[4] developed a personalized news article recommendation system by exploiting contextual information about the users and articles. Rafael Sotelo, Yolanda Blanco-Fernández, Martín López-Nores, Alberto Gil-Solla and José J. Pazos-Arias[5] proposed a TV program recommendation approach, which created personalized TV schedules for groups of users based on TV-Anytime descriptions of TV contents and semantic reasoning techniques. Our team is inspired by the idea of recommendation using classification, and we will implement a feature-based multinomial classification system to give job seekers best recommendation.

3. Dataset and Features

3.1 Data Collection

As mentioned before, we collect employee data from LinkedIn website. We have gathered 609 pieces of information and divided the samples into a training set of 450 samples and a test set of 159 samples. In particular, for the training set, we have 150 samples from Google, 150 from Apple and 150 from Facebook. For the test set, we have 53 samples from Google, 53 from Apple, and 53 from Facebook. The sample split is random, and later we used cross validation to verify the result.

3.2 Features

We first consider an employee's industrial experience like years in industry, years in the current company and past internship experiences. These features are reasonable indicators of a person's past working experience. Secondly, we explore feature related to the academic attributes of employees like the highest degree and academic publication. Personal skills are also taken into account. There are features like the number of skills listed LinkedIn homepage and the number of the endorsement of the most endorsed skill. Finally, we consider some personalized attributes like gender and the employee's linguistic skills.

In summary, the features we are experimenting on right now include whether an employee has more than 1 or 5 or 10 year(s) working experience, whether he/she has more than 1 or 5 year(s) of working in the current company, whether he/she has a Doctoral/Master's degree, whether he/she has publications or patents and the number of publications and patents, whether he/she is multilingual, whether the number of most endorsed skills is greater than twenty, whether he/she has internship experiences in each of the three companies, and finally the gender of an employee.

4. Methods

4.1 Labeling

There are several ways to give labels to the training set. For Naive Bayes and Decision Tree, the labeling is not important. So we directly label the companies as 0,1,2...k-1, given k is the number of companies.
For SVMs, we decompose the multiclass classification problem into binary classifications. We generally have 3 decomposition methods including one-to-rest, one-to-one, and an algorithm called error-correcting-output-codes. For each of the three methods, we need to build k, k*(k-1)/2, and n (code word length n) classifiers respectively.
For Neural Network, we use one-hot labeling, where Google is [1 0 0], Facebook is [0 1 0] and Apple is [0 0 1].

4.2 Models

4.2.1 Decision Tree

One big problem with using decision tree model is that it will definitely have overfitting on the training set. However, we can still get some good insight on training data, which includes whether the training set is linearly separable, and how useful a feature is in classification.

4.2.2 Naive Bayes (with smoothing)

We start from the Naive Bayes classifier we have implemented in problem set 2, and add one more dimension to the labels. We also included smoothing to achieve a better result.

4.2.3 SVM (linear kernel and Gaussian kernel)

We assume that the input data is linearly separable and build binary linear SVM classifiers. For the multi-class problem, we adopt all of the three decomposition methods.

One-to-rest decomposition: we build three binary classifiers. For each classifier, the label is 1 for interested company (like Facebook), -1 otherwise.

One-to-one decomposition: we build three binary classifiers including Google vs. Facebook, Google vs. Apple, and Apple vs. Facebook. Label one of the companies as 1, rest -1.

Error-correcting-output-codes (ECOC): we choose a code length of 2 and build 2 binary classifiers. For details of the ECOC algorithm, please refer to online documents.

We also use SVM classifier with Gaussian kernel for comparison.

4.2.4 Neural Network

We created the neural network with 1 hidden layer and 2 hidden layers. The input layer has dimension of 14*1. The size of each hidden layer is 100. The function on hidden layer is hyperbolic function (tanh). The output layer has dimension 3*1. Finally we use a softmax function to produce the final one-hot label. The loss function is the negation of log-likelihood plus a regularization factor. We used L1 regularization with lambda equal to 0.0001.

5. Experiments and Results

5.1 F1 scores using different methods

For Naive Bayes, we build our own model using MATLAB. We utilized the Python Scikit-learn packages to test the result of SVM and Decision Tree. The Neural Network is built using Java. The results of these algorithms are shown in the table 1.

Model	Decision Tree	Naive Bayes (with smoothing)	Support Vector Machine	Neural Network (1 hidden layer)	Neural Network (2 hidden layers)
F1 score	50.94%	58.49%	63.52%	66.04%	66.04%

Table 1: F1 score of different models

As expected, Decision Tree works worst, since it has overfitting problem on the training set.

For Naive Bayes, the result without smoothing is worse than SVM and Neural Network. However, if we eliminate some features, we can get a result of 62.26%, which is slightly worse than SVM.

For SVM, the linear kernel SVM with ECOC or 1-v-rest decomposition works best and achieves an F1 score of 63.52%, which slightly outperforms the other decomposition scheme (1-v-1, with a F1 score of 62.26%). The three decomposition methods do not make a big difference to results, so we built another linear kernel classifier without decomposition methods and found a result of 63.52%, which is same as the best result we have obtained. When we build the Gaussian kernel classifier, we again did not use any decomposition method. It gives a result of 59.11%, which is worse than the linear kernel classification results.

For Neural Network, since there could be some local maximum, we have decided to train the model multiple times with random weight initialization and choose the one with best result (minimal loss). By tuning parameters, we have found that the best learning rate is 0.001. The best number of iterations is 50 for one hidden layer, 30 for two hidden layers. The number of hidden layer and the number of items in hidden layers do not affect the result.

In addition, we have discussed about the findings on Facebook in milestone report, where none of the classifier can label Facebook employees correctly. The further investigation of this phenomenon is included in the next section.

5.2 Confusion Matrix

Test set			
Company	Precision	Recall	F1 score
Google	56.25%	84.91%	67.67%
Facebook	80.00%	37.74%	51.28%
Apple	74.07%	75.47%	74.77%
Train set			
Company	Precision	Recall	F1 score
Google	56.19%	72.67%	63.37%
Facebook	77.68%	58.00%	66.41%
Apple	64.58%	62.00%	63.27%

Table 2: confusion matrix from neural network on train set and test set

The confusion matrix in the table 2 illustrated some interesting phenomenon on Google and Facebook. Similar patterns are observed not only in neural network, but in other models as well. We can see Google has low precision and high recall, while Facebook has high precision and low recall. This means we tend to correctly classify Google employees, but we also include employees from other companies. For Facebook, we are pretty confident with employees labeled with Facebook, but we still classify true Facebook employees to other company. There could be several reasons for this behavior. Maybe some of Facebook employees share a strong pattern. It is also possible that Google does have a diverse employee body, so it is hard to tell who is real Google employee. In the future we may further investigate this behavior by collecting more data.

5.3 Feature Analysis

5.3.1 Feature effectiveness analysis

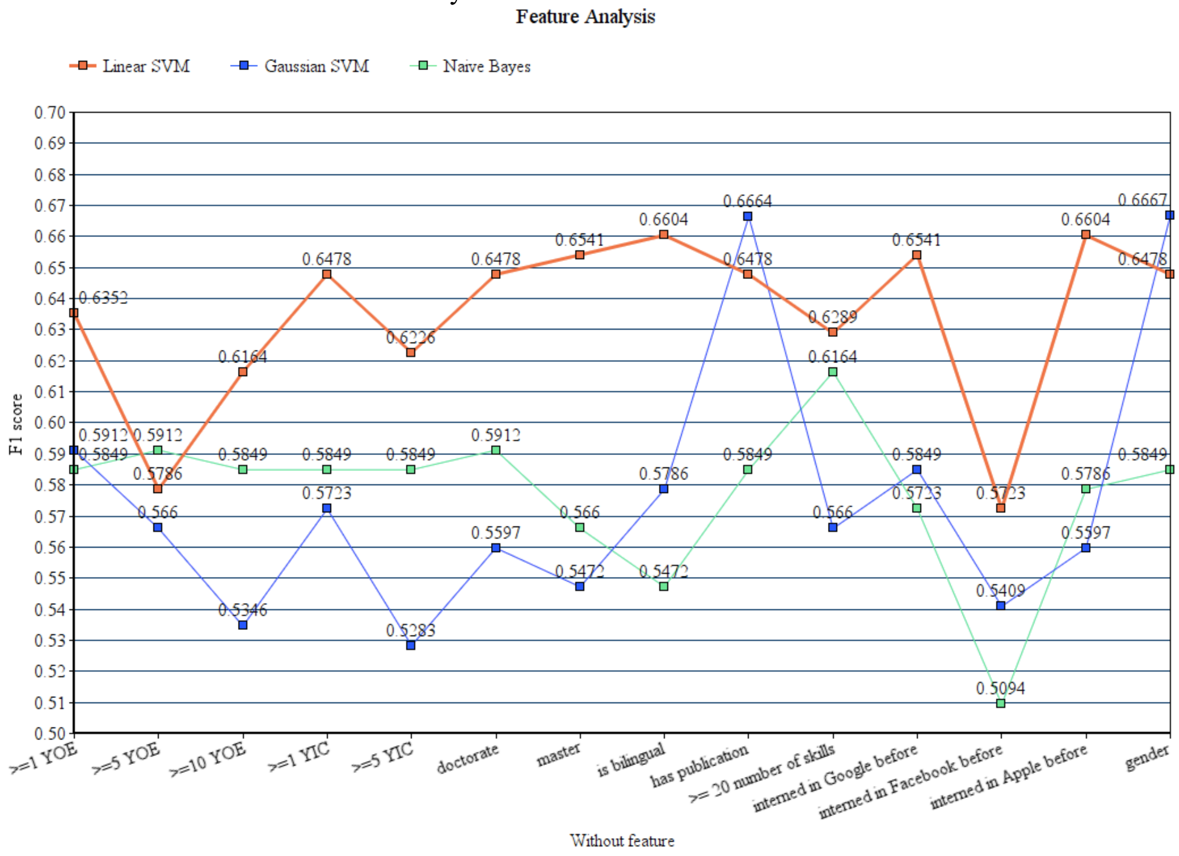

Figure 1: Feature analysis across different methods

We tested feature effectiveness by excluding one feature and see the result change in different models. The results are shown in Figure XX. We can see that each model has its own ineffective features, and it is hard to identify any feature that is not effective for all models. On the other hand, if we exclude the feature of Facebook internship, we can see a significant f1 score drop in all models. It is obvious that Facebook internship is the most effective feature.

5.3.2 Feature weight examination

We have examined the weight of each feature on each company in Naive Bayes model, and we have some interesting findings.

Apple has more employees with long years of industrial experience. There are more long time employees in Apple than the other two companies. Also Apple favors people with many skills.

Google has the most employees who just joined the company. There are also more Google employees with master's degree and bilingual/multilingual abilities.

Facebook has most employees who have been in the company for 1 to 5 years.

Finally, internship experience plays a significant role in classifier. The companies all have a significant percentage of employees who have former internship experience in current company.

5.4 4-Fold Cross Validation

At the final stage, 4-fold cross validation is applied to help evaluate different models. We split all the 609 data samples into 4 disjoint subsets randomly, repeatedly trained on 3 of the subsets and then tested on the left one subset.

5.4.1 SVM

For the SVM with linear kernel and Gaussian kernel, following result is obtained. It is easier to see that the F1 score of SVM using cross validation is similar to the result in section 5.1.

Test Set	1	2	3	4	Average
Test F1 score (linear)	64.71%	66.01%	64.47%	56.95%	63.04%
Test F1 score (Gaussian)	58.82%	56.21%	60.53%	52.98%	57.13%

Table 3: F1 score of SVM with cross validation

5.4.2 Neural Network

Test set	1	2	3	4	Average
Test F1 score	66.04%	50.00%	72.67%	60.00%	62.18%
Train F1 score	64.22%	68.63%	62.53%	64.49%	64.97%

Table 4: F1 score of Neural Network with cross validation

Since the average f1 score on training set is pretty close to the average f1 score on test set, we are confident that our implementation of neural network will not overfit the train set, and the result is quite close to the to the generalized accuracy.

6. Conclusions and Future Work

Our team is pretty happy with our result. Using either SVM or Neural Network, we are able to achieve F1 scores around 63% on test set. Since training error is close to test error, our models do not overfit train set and is a good estimate of the generalized error.

Also, our team has found some interesting features that different companies looked for. Before this project, we have heard some rumors, such as master's degree is important to get in Google, and Apple favors experienced employees. We are glad to have proved these claims after the project.

In addition, our model recommended Google to all of our team, which does support our finding that Google includes everyone. The possible future works include expanding the number of companies and include more features to receive a more diverse recommendation system.

References

[1] Douglas Eck, Thierry Bertin-Mahieux, Paul Lamere, and Stephen Green, Automatic generation of social tags for music recommendation. *Neural Information Processing Systems Conference (NIPS)* 20. Vancouver, British
[2]Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli, Alfonso Miola, and Giuseppe Sansonetti, Context-aware movie recommendation based on signal processing and machine learning, *Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, p.5-10, October 23-27, 2011, Chicago, Illinois
[3]Yukun Cao, and Yunfeng Li, An intelligent fuzzy-based recommendation system for consumer electronic products, *Expert Systems with Applications: An International Journal*, v.33 n.1, p.230-240, July, 2007
[4] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the Nineteenth International Conference on World Wide Web, 2010
[5] Rafael Sotelo, Yolanda Blanco-Fernández, Martín López-Nores, Alberto Gil-Solla, and José J. Pazos-Arias, “TV program recommendation for groups based on muldimensional TV-anytime classifications,” IEEE Trans. Consum. Electron., vol. 55, no. 1, pp. 248-256, Feb. 2009