Resume Screening System Based on Machine Learning

Shengcheng Chen, Shunqi Liu, Zhuofan Tang, Tianyi Wu

Abstract

With no job experience, newly graduates from schools need to know if their backgrounds make them suitable for a certain job. Meanwhile, recruiters need assistance to determine who can enter the next round out of many job applicants. In this paper, we use several machine learning classifiers and fuzzy labeling Naive Bayes method to determine if a job seeker is suitable for a certain job title. We obtain people's LinkedIn profiles and background information including education, internship and skill sets. In the training, we use people's most recent experience as labels. For a recent graduate, we use the model to predict if he or she fits a certain job based on the background. The result can be used for HR to help determine if one can pass screening and help applicants to evaluate their own chance of success in a job application.

1 Introduction

Job hunting can be a tedious process for both job seekers and recruiters. On one hand, dealing with huge amount of job application, recruiters spend hours to look at applicants' profiles and determine if they are qualified for the second round. On the other hand, many newly graduates do not know what job they want to seek for and the chance of success on application for a certain job title. The goal of this project is to run machine learning models and help both recruiters and job seekers evaluate if a certain job title is suitable for a person given his or her backgrounds. The result of the paper is that we achieve accuracies that are at least 80%, which is much larger than the baseline accuracy. Also, the fuzzy label Bayes gives both confidence and predictions successfully.

2 Literature Review

Basing on the screening problem, Harsh Jain et al. [1] consider a way to generate word vectors, then build machine learning models based on those vectors, which mostly comparing the distance of the vectors. However, under this method, some useless contents may be included in the model, which makes error on the prediction. To solve this problem, we extract key words of each profile instead of including all the words. Furthermore, inspired by Yangchuan Tang et al. [2] and G. Yazgı Tütüncü et al. [3], who use fuzzy Bayes train the model, we consider adding fuzzy labels, which gives confidence of the prediction and help recruiters making judgements about the profile.

3 Problem Definition and Algorithms

3.1 Task Definition

Initially, we are trying to recommend the appropriate job based on one's LinkedIn profile. However, the task is beyond our reach because of several aspects. First, it is a multi-classification problem that we are not very familiar with. Then, it is impossible to generalize the entire dataset to a bunch of job titles and they might not be enough data/distinction for each category.

Thus, to simplify and explore more based on our current knowledge, we translate this problem into a binary classification problem and show its possible application in Corporation Hiring Process.

1). We specify a job title for the model. In our example, we used 'Product Manager' job class.

2). We process our training sets' labels accordingly. 1 for 'Product Manager' and 0 for not.

3). We use our trained model to predict on profile and get a prediction of whether this profile is a good fit for the specified job title or not.

3.2 Algorithm(s) Definition

In order to deal with words and documents, we use a well-known package gensim and mainly implement Word2Vec model and similarity model. Word2Vec is a recent model that embeds words in a lower-dimensional vector space using a shallow neural network. The result is a set of word-vectors where vectors close together in vector space have similar meanings based on context, and word-vectors distant to each other have differing meanings. Similarity model is used to compute the similarity between two words which help to classify and define titles and skills in resume.

3.3 Expectations

As we will establish a baseline method and a bottom line for accuracy. We hope that our machine learning classifiers will perform much better than the baseline model not only in accuracy but also in the robustness to noise.

4 Methodology

In this section, we show in detail how we intuitively construct the baseline model for comparison and how we run through different machine learning classifiers to find the optimal model.

4.1 Dataset Preparation

For our training dataset, we used a LinkedIn dataset which consists of 100000 people's linkedin profile. For each person, we selected the following attributes as similar literature has suggested (Jain and Kakkar, 2019) [1]: locality, industry, summary, specialties, interests, major, education, degree, affiliation. For the ease of our training, we dropped entries with missing values and the result is that for our final training set, we have 5701 entries with the above values. We selected people with their latest experience' title as raw label.

4.2 Label Processing

As the labels extracted from data are characters of job titles, we want to transfer them to {0,1} labels. One way to do that is to compare the similarity of target job title and label job titles.

The labeling rule is shown as Eq. (1):

$$label = \begin{cases} 0 & similarity < 50\% \\ 1 & similarity \geq 50\% \end{cases} \quad (1),$$

where label is binary labels, similarity is percentage.

Calculation of similarity is based on build in function of gensim.model.similarity. Also, the models in gensim are trained by word2vec and the text from raw data, [4], [5] and [6].

4.3 Baseline Method

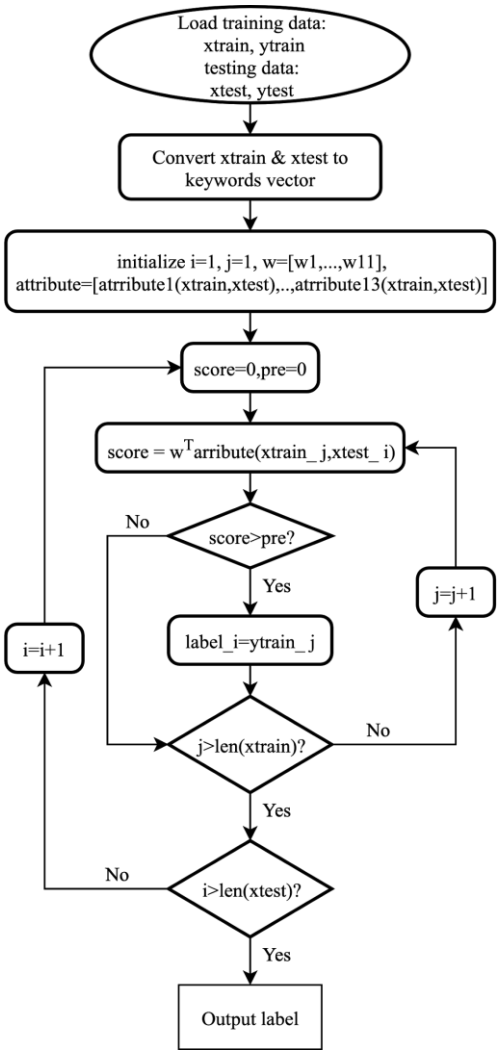

Figure 1: Flow chart of baseline model.

Baseline is used for comparing the accuracy of non-machine learning method and machine learning method. To give the reason why we

should use machine learning to solve this problem.

Our baseline is based on the similarity of keywords in each attribute, like position, degree, etc. We firstly compute the score, which is weight sum of each attribute, of each testing data with all labeled data (training data). Then assign the highest score training data label (job title) to the testing data. Finally compute the accuracy. The flow chart is shown in Figure 1.

Here w is the weight on each attribute, and attribute is vector contains the similarity of keyword of *xtrain* and *xtest* at *n*-th attribute.
After output the job title labels, we should convert them to the {0,1} labels based on the method in 3.2. Then compute the accuracy using predicted {0,1} labels and testing labels.

4.4 Machine Learning Classifier

As suggested above, we tried to beat the baseline model using selected machine learning classifiers. Prior to each model, we run a preliminary test for hyperparameter selection.

For each classifier, we reckon that the amount of useful information is related to the number of keys we used to featurelize, shown in Figure 2. Also, we want to explore how many instances we need to achieve a reasonable accuracy. Thus, we sweep number of keys included and the number of instances included for each classifier to get an accuracy overview ([50, 100, 250, 500, 1000, 2500, 5000]), as shown in Figure 4 to Figure 13.

4.5 Fuzzy Label Naive-Bayes

As we generate {0,1} labels with uncertainty, it is a better way to maintain the probabilities of labels, which we call them confidence. We retrain the naive Bayes classifier with those fuzzy labels, by considering the P(y) and P($v|y$) are changed by the confidence of labels, we build a new naive Bayes model.

In prediction part, we use the new model to do prediction on testing data, then compute the accuracy by method using in 3.3. Furthermore, the accuracy here can be used as the confidence of the model, which means the prediction based on this model has some probability of correctness. This probability of correctness can give advice for recruiter about the prediction.

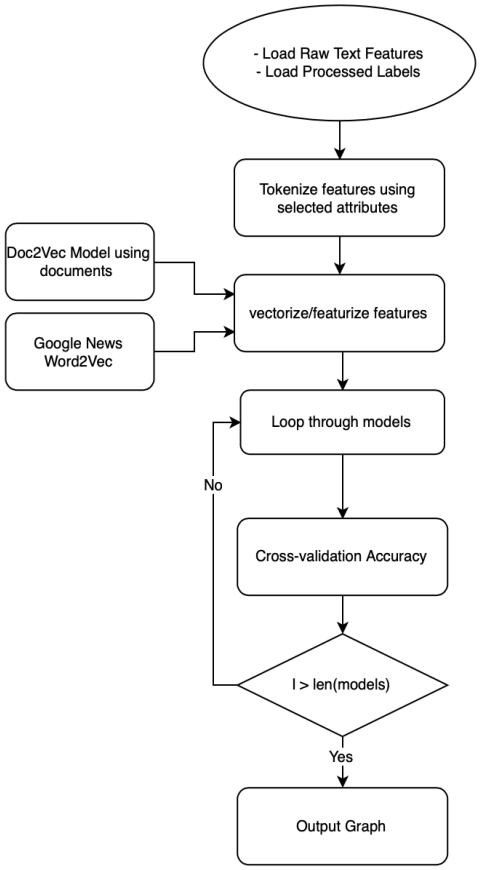

Figure 2: Model graphing.

5 Result

5.1 Baseline Method

For the baseline method, we run the model on random 4000 training data and test on the rest 1701 testing data for 5 times, the average accuracy of this method is around 65%.

5.2 Machine Learning Classifiers

After running the models on feature and cross-validated using k-fold, the results are presented as follows. For Figure 3, the overall accuracy comparison used accuracies trained using the optimal hyperparameter for each model compared with the baseline. For Figure 2 through Figure 7, cross-validated accuracies for each model trained are plotted with different numbers of instances.

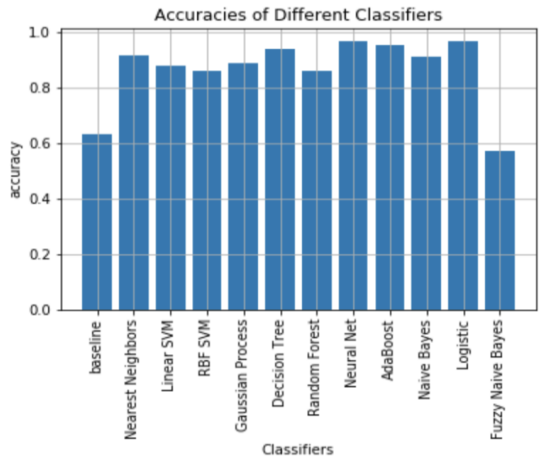

Figure 3: Overall Accuracy Comparison.

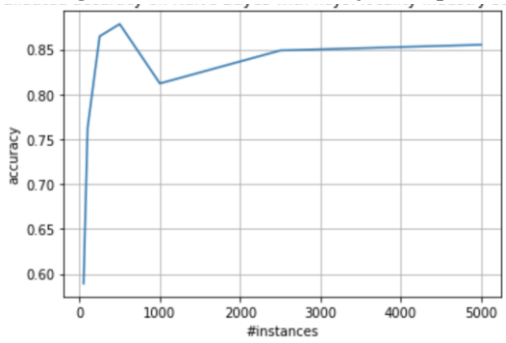

Figure 4: Naive Bayes.

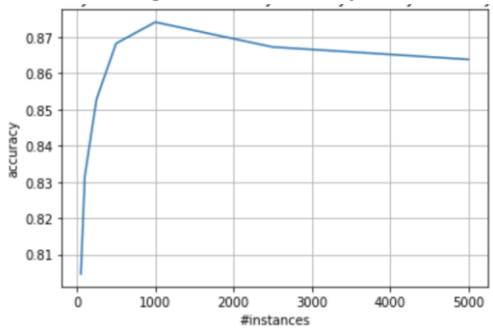

Figure 5: Linear SVM.

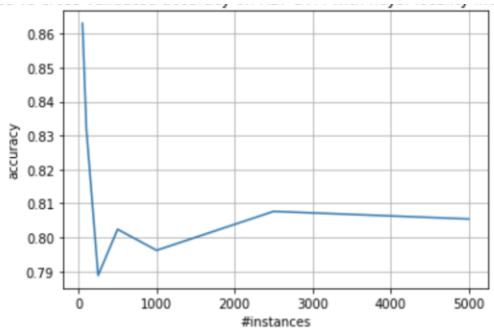

Figure 6: RBF SVM.

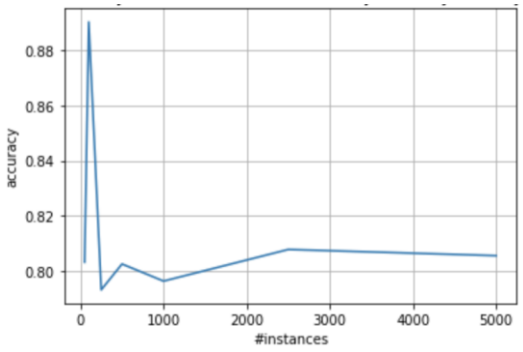

Figure 7: Linear Regression.

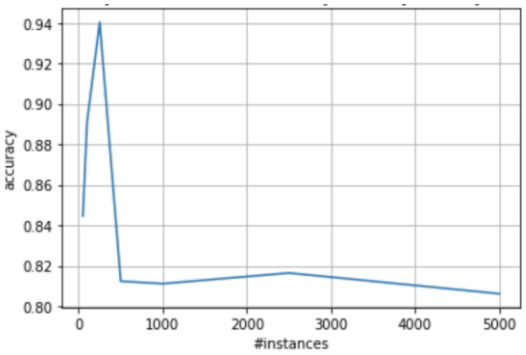

Figure 8: Decision Tree.

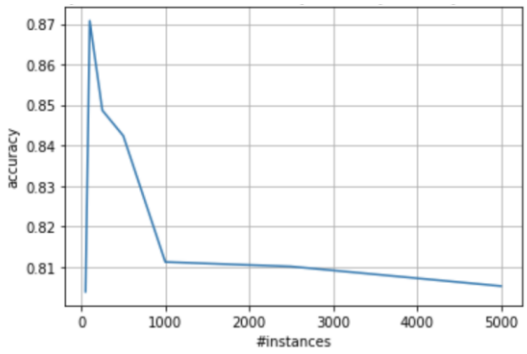

Figure 9: Random Forest.

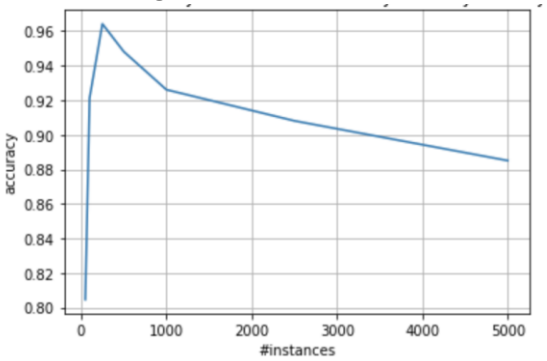

Figure 10: Neural Network.

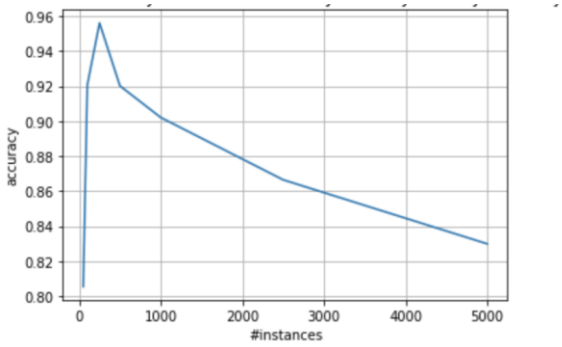

Figure 11: Adaboost.

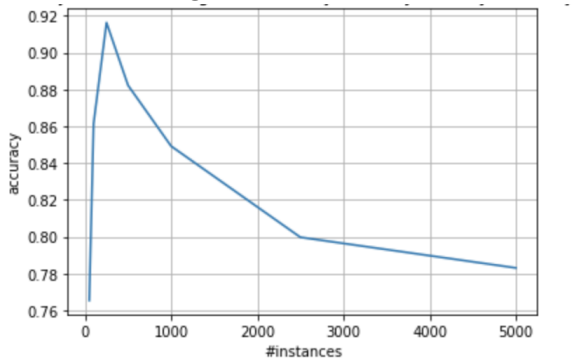

Figure 12: K-NN.

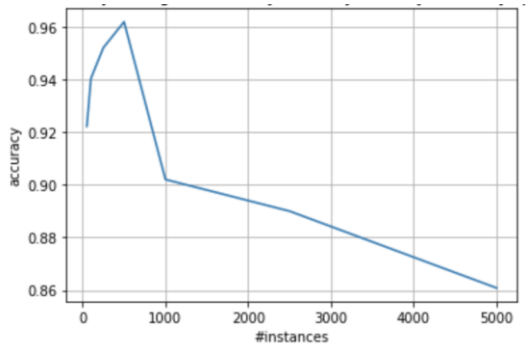

Figure 13: Logistic.

5.3 Fuzzy Label Naive-Bayes

Randomly select 4000 training data for 5 times, to construct 5 fuzzy label naive Bayes classifier. then test the accuracy on the whole dataset. The average accuracy (confidence) is 57%, which means using this model we have 57% probability to predict the true label correctly.

6 Discussion

6.1 Baseline

According to Fig. 3, the accuracy of baseline is worst, comparing with other machine learning methods. It shows that the machine learning method has better performance on this problem than the directly compare the keywords similarity.

6.2 Machine Learning Classifiers

6.2.1 Accuracy Overview

As shown in the above graphs, the overall accuracies of the machine learning classifiers perform better than intuitive baseline method above 80% accuracy for all classifiers after training on 5000 instances. Out of all the classifiers, Neural Network exhibit the best overall accuracy and robustness to noise and future data (92%-97% mean accuracy of 5-fold cross validation on 5000 instances).

6.2.2 Patterns

For all classifiers, we notice a common pattern that first the accuracy improved drastically from 50 instances to 250 instance and then drop as more instances added. We dig into this pattern and suspect that at first, the model quickly goes from underfitting to overfitting. Then as more instances added, it gradually generalizes itself for the population.

7 Future Work

Due to the limit of our knowledge and the amount of time we can invest in this project for our translation from multi-classification to binary classification, we simplified the situation. If more time were given, we can expand our model to do multi-classification of a class of job titles and select the *Argmax* of them.

8 Conclusion

After implementing all the machine learning classifiers above, we achieve accuracies that are at least 80% which is much larger than the baseline accuracy. Among these methods, Neural Network performs best with around 95% accuracy and robustness to noise. Besides, we introduce fuzzy label Naïve-Bayes as an extra criterion for judgement. Fuzzy label Naïve-Bayes provides average confidence for whole database and evaluating how good the label and features we constructed which is 57% in our case. In

conclusion, the Resume screening system we built based on machine learning can help HR filtering job seekers at the very first moment without necessity to look through every resume. It can also help job hunters assessing themselves before they submit their resumes.

References

[1] Harsh Jain and Misha Kakkar. 2019. *Job Recommendation System based on Machine Learning and Data Mining Techniques using RESTful API and Android IDE*. In International Conference on Cloud Computing, Data Science & Engineering (Confluence), pages 416-421.
http://ieeexplore.ieee.org.proxy.library.upenn.edu/stamp/stamp.jsp?tp=&arnumber=8776964&isnumber=8776606

[2] Yongchuan Tang and Yang Xu. 2005. *Application of fuzzy Naive Bayes and a real-valued genetic algorithm in identification of fuzzy model*. Information Sciences,Volume 169, Issues 3–4, pages 205-226.
https://doi.org/10.1016/j.ins.2004.05.004.

[3] G. Yazgı Tütüncü and Necla Kayaalp. 2015. *An Aggregated Fuzzy Naive Bayes Data Classifier*. Journal of Computational and Applied Mathematics, Volume 286, pages 17-27.
https://doi.org/10.1016/j.cam.2015.02.004.

[4] Google Code. 2013. *Tool for computing continuous distributed representations of words*.
http://code.google.com/archive/p/word2vec/

[5] Rachael Tatman. 2018. *How semantically similar are two job titles.*
https://www.kaggle.com/rtatman/how-semantically-similar-are-two-job-titles

[6] Mad Hab. 2017. *Online Job Postings : Dataset of 19,000 online job posts from 2004 to 2015*.
https://www.kaggle.com/madhab/jobposts