Multi-Label Classification for Tagging Job Descriptions

Luke Lefebure

April 6, 2017

1 Introduction

This report details my approach to Indeed's Machine Learning competition hosted by Hackerrank. See here for more details. The task is a multi-label classification problem that involves assigning tags such as "full-time-job" to job descriptions.

2 Data

The data is provided by Indeed and consists of a training set of job descriptions labeled with a variable number of tags and a test set of job descriptions without labels. The training set consists of 4,375 samples, and the test set has 2,921. A full list of tags along with their frequency in the training set is in the table below.

Tag	**Frequency**
1-year-experience-needed	331
2-4-years-experience-needed	1043
5-plus-years-experience-needed	636
associate-needed	209
bs-degree-needed	970
full-time-job	885
hourly-wage	451
licence-needed	524
ms-or-phd-needed	83
part-time-job	328
salary	669
supervising-job	751

2.1 Training Data Quality

There are 871 samples in the training data that have no associated tags. In several cases, this seems to be a result of error in the original tagging process. For example, 33 of those descriptions contain the string "full time".

2.2 Labels

As mentioned in the problem description, there are several tags that do not occur together. From analyzing their cooccurrence, I see that the following tuples are mutually exclusive:

1. '1-year-experience-needed', '2-4-years-experience-neeed', '5-plus-years-experience-needed'
2. 'bs-degree-needed', 'associate-needed', 'ms-or-pd-needed', 'license-needed'
3. 'salary', 'hourly-wage'
4. 'full-time-job', 'part-time-job'

3 Feature Extraction

I hypothesize that the most important features for this task are the presence of a limited number of keywords. For example, I expect samples tagged with "hourly-wage" to contain substrings like "per hour" or "hourly". Therefore, my process for generating features is as follows:

1. Generate binary document-term matrix for the training set using unigrams, bigrams, and trigrams as terms.
2. Generate binary matrix representation of training label set.
3. Discover the terms most highly correlated with each tag by calculating Pearson's R between each column in the two matrices above.
4. Reduce the dimension of the document-term matrix to include only terms that are in the top 15 most correlated for some tag. This matrix is what I use as input to my classifiers.
5. Build binary document-term matrix for test set using vocabulary defined in the step above.

The result of step three above seems to prove my hypothesis. Below are the top three correlated terms with each tag:

Tag	**Term 1**	**Term 2**	**Term 3**
1-year-experience-needed	1 year	year	1
2-4-years-experience-needed	years	2 years	3 years
5-plus-years-experience-needed	5 years	5	years
associate-needed	associates degree	associates	associates degree or
bs-degree-needed	degree	bachelors	bachelors degree
full-time-job	full-time	full time	a full-time
hourly-wage	hour	per hour	hourly
licence-needed	nurse	rn	licensed
ms-or-phd-needed	masters	masters degree	masters degree in
part-time-job	part time	part-time	a part time
salary	salary	competitive salary	benefits
supervising-job	manager	leadership	management

3.1 Notes

1. I choose the set of the top 15 most correlated terms with any tag as my vocabulary because this number produced the best results through experimentation.
2. I do not do any stopword removal because if a term is highly correlated with a tag, it does not matter if it is a stopword. Additionally, words like "one" and "two" are commonly considered stopwords, but those are very relevant to this task.
3. I do not do any stemming. The term "associate" could be used in the context of "seeking sales associate", or it could be used as "associates degree required". The stem can be very relevant for this task.
4. My input features are highly correlated by the nature of their construction. If I had more time, I would investigate this problem further.

4 Methods

4.1 Baseline

I consider two simple baselines that involve predicting the most common tags in the training set.

1. **PopularTag1**: predict 2-4-years-experience-needed for all samples
2. **PopularTag4**: predict 2-4-years-experience-needed, bs-degree-needed, full-time-job, and supervising-job for all samples

4.2 Keyword Model

I consider a simple keyword rule for predicting tags. This involves predicting a tag if the unigram, bigram, or trigram with the highest correlation with that tag is present. For example, since the unigram "full-time" is most highly correlated with the tag "full-time-job", I predict that a description is tagged with "full-time-job" if it contains the substring "full-time".

4.3 SVM and Logistic Regression

These methods produce linear decision boundaries and consequently achieve similar scores. To generalize to the multi-label problem, both work by constructing a binary classifier for each tag in a "OneVsRest" scheme.

Both methods produce probability estimates for a specific tag belonging to a certain sample. By default, both will predict a tag if the probability estimate for that tag is greater than .5. However, for both methods, this caused a significant imbalance between the precision and recall. In particular, precision was much higher which shows that the models fail to predict many tags but do relatively well when they do predict a tag. This means that the models could likely benefit from predicting more tags to boost the true positive rate at the expense of increasing the false positive rate.

By varying the probability threshold at which a tag is predicted to belong, I can tune the tradeoff between precision and recall. For SVM, the optimal threshold was around .24, and for logistic regression, it was around .3.

5 Results

5.1 Evaluation Metric

Predictions are scored using the F1 metric. In the case of multi-label classification, true positives/negatives and false positives/negatives are counted by summing over all tags.

5.2 Scores

The table below shows results from my different models. The scores reported are those that were achieved from using the best tuning parameters found through experimentation. I report scores on a 30% held out validation set.

Method	Training Score	Val Score	Val Precision	Val Recall	LB Score
PopularTag1	.185	-	-	-	-
PopularTag4	.299	-	-	-	-
TopCorrelatedKeyword	.358	-	-	-	-
OneVsRest Linear SVM	.718	.665	.763	.589	-
OneVsRest Linear SVM w/ threshold	.740	.686	.685	.687	681.37
OneVsRest Logistic Reg w/ threshold	.736	.693	.692	.693	676.29

5.3 Final Prediction

My final submission is the OneVsRest Linear SVM w/ threshold model.

6 Conclusion

Because my time on this project was limited, there are many shortcomings:

1. As mentioned above, my input features are highly correlated and more work could be done on engineering better features.
2. The mutual exclusivity constraints on the tags detailed earlier are not enforced by the models. I attempted to prune predictions to abide by these constraints, but that did not improve the score. This is something that I would have liked to look at further.
3. I only looked at classifiers with linear decision boundaries and did not consider interaction between features. Introducing more complexity to the models is another thing that I would have liked to try.

4. The training set had many descriptions with no tags, many of which seemed to be in error. I would have liked to explore the effect of dropping these samples while training.

5. I could do a better job of evaluating my models by using cross validation instead of a simple validation set.