iNetworks task

Recommendation of Job Functions For Specific Job title

By : Khaled Madkour
Gmail: khaled.madkour@gmail.com
Linkedin: https://www.linkedin.com/in/khaled-madkour/
Github: https://github.com/KhaledMadkour

PROBLEM STATEMENT

Job functions and job titles are very different things. A job title is essentially the name of a position within an organization filled by an employee. Job function is the routine set of tasks or activities undertaken by a person in that position. An employee's title and function are often closely related, though not all job functions are clear based on the title alone.

The mission is to build a system that recommends the job functions that an employee with a job title can perform.

Data overview

The dataset has 3 columns each referring to (index, Title, Job Function, Industry) respectively. We have exactly 10,800 rows.

After some data processing I found a couple of important remarks. Job Function is a string of one or more job functions corresponding to certain job title. Also After sorting according to job title I noticed that multiple job titles may exist in our dataset each referring to either the same or different job functions than each other.

Those speculations Made it clear that I am going to use a multilabel classifier as the dataset has multiple labels (job functions) for each title.

```
df = df.ix[:,(1,2)]
print (df.sort_values(by=['title']))
```

```
                                    title                                        jobFunction
9149                  .NET Backend Developer  ['Engineering - Telecom/Technology', 'IT/Softw...
1045                  .NET Backend Developer  ['Engineering - Telecom/Technology', 'IT/Softw...
9413                  .NET Backend Developer  ['IT/Software Development', 'Engineering - Tel...
3694   .NET Core Developer - Senior\Team Lead  ['IT/Software Development', 'Engineering - Tel...
5263                           .NET Developer  ['Engineering - Telecom/Technology', 'IT/Softw...
...                                      ...                                              ...
7055         موظفة نشر و سوشيال ميديا اونلاين  ['Media/Journalism/Publishing', 'Marketing/PR/...
10171        موظفة نشر و سوشيال ميديا اونلاين  ['Media/Journalism/Publishing', 'Marketing/PR/...
1668         موظفة نشر و سوشيال ميديا اونلاين  ['Media/Journalism/Publishing', 'Marketing/PR/...
8514         موظفة نشر و سوشيال ميديا اونلاين  ['Marketing/PR/Advertising', 'Media/Journalism...
9303         موظفة نشر و سوشيال ميديا اونلاين  ['Marketing/PR/Advertising', 'Media/Journalism...
```

Answering Questions:

1 Data Cleaning

1.1 Job Titles

Job titles are our only feature in the model. To get better accuracy. I need to clean those titles a little. So I performed certain functions on this list.

-Used Stemmers , they remove morphological affixes from words, leaving only the word stem. Using ntkl library.
-Removed any stop words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that should be ignored as they don’t provide any meaning or use in our title.
-Removed special characters.
-Changed all titles to lowercase letters.

```
stemmer = PorterStemmer()
words = stopwords.words("english")
df['title'].apply(lambda x: " ".join([stemmer.stem(i) for i in
re.sub("[^a-zA-Z]", " ", x).split() if i not in words]).lower())
```

1.2 Job Function

Job functions are provided as strings in our dataset. So they had to be changed into arrays. That was done using “ast.literal_eval” Function.

```
df['jobFunction'].apply(ast.literal_eval).apply(np.sort)
```

As I previously pointed out. Multiple data with the same job title exists in the data frame. So I grouped all job functions with job title to make list of lists of all job functions for the same title.

```
df.groupby('title').jobFunction.apply(list).reset_index()
```

Then I had to flatten the job Functions while removing any duplicates, also removing special characters using Custom method Called “Flatten”.

```
df['jobFunction'].apply(flatten)
```

At last, I removed all rows that has either no titles or job functions.

```
df[df['jobFunction'].map(lambda d: len(d)) > 0].reset_index()
df[df['title'].map(lambda d: len(d)) > 0].reset_index()
Clean_df.head()
```

title	jobFunction
abap consult	[EngineeringTelecomTechnology, ITSoftwareDevel...
account	[AccountingFinance, Administration, HumanResou...
account account payabl	[AccountingFinance]
account account receiv	[AccountingFinance]
account alexandria	[AccountingFinance]

2 How does the model work?

I used the binary relevance method with naive bayes classifier for my model. So let's dive a little deeper into that.

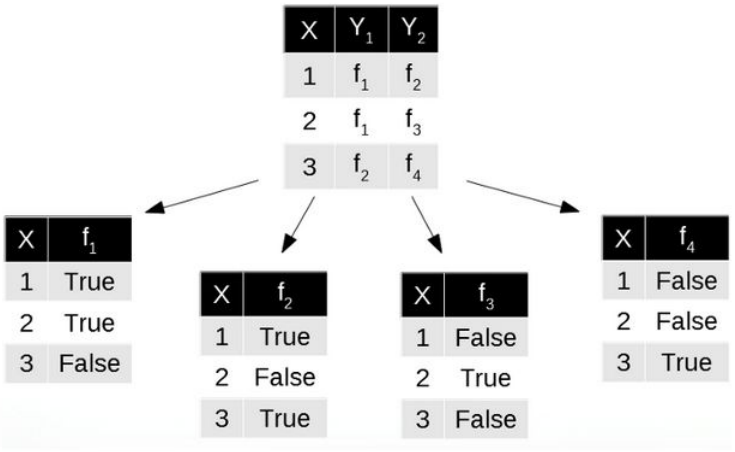
2.1 Binary Relevance

Binary relevance The binary relevance method (BR) is the simplest problem transformation method. BR learns a binary classifier for each label. Each classifier C1 , . . . , Cm is responsible for predicting the relevance of their corresponding label by a 0/1 prediction:

$$C_k : \mathcal{X} \longrightarrow \{0,1\}, \quad k = 1,\ldots,m$$

These binary predictions are then combined to a multilabel target. An unlabeled observation x(l) is assigned the prediction $\left(C_1\left(\mathbf{x}^{(l)}\right), C_2\left(\mathbf{x}^{(l)}\right), \ldots, C_m\left(\mathbf{x}^{(l)}\right)\right)^{\top}$

Hence, labels are predicted independently of each other and label dependencies are not taken into account. BR has linear computational complexity with respect to the number of labels and can easily be parallelized.



Binary Relevance example

Binary Relevance uses Naive Bayes classifier.

2.2 Naive Bayes

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of A happening, given that B has occurred. Here, B is the evidence and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Here in our problem we have problem of classifying title with certain job functions. As discussed Binary Relevance split our labels making each one act as independent identifier. So we are going to explain on only one job function. So in our problem we are trying to guess whether a title is masked to a certain specific job function let this job function be (Software development). The equation will be as follows: $P(y|X) = \frac{P(X|y)P(y)}{P(X)}$

So The variable y is the class variable(Software development), which represents if it is job function is Software development given a certain title. Variable X represent the parameters/features (job title).

2.3 MultiLabelBinarizer

Job functions exist in the dataset as an array for each title. To use binary relevance I need to create new columns each corresponds do certain job function(certain class/label). And for each title, there is a (1) value at each job function the title can perform.

```
This was achieved using MultiLabelBinarizer.
multilabel_binarizer = MultiLabelBinarizer()
multilabel_binarizer.fit_transform(Clean_df['jobFunction'])
y = multilabel_binarizer.transform(Clean_df['jobFunction'])
for idx, job in enumerate(multilabel_binarizer.classes_):
  Clean_df[job] = y[:,idx]
Clean_df.head()
```

title	jobFunction	AccountingFinance	Administration	AnalystResearch	Banking	BusinessDevelopment
abap consult	[EngineeringTelecomTechnology, ITSoftwareDevel...	0	0	0	0	0
account	[AccountingFinance, Administration, HumanResou...	1	1	0	0	0
account account payabl	[AccountingFinance]	1	0	0	0	0
account account receiv	[AccountingFinance]	1	0	0	0	0
account alexandria	[AccountingFinance]	1	0	0	0	0

We later split the data into 80% training and 20% test.

3 Why did I choose this approach ?

At first I noticed the dataset contains multiple labels for each title. So naturally this is a Multi-label classification problem. In my code I tried different methods and algorithms and compared between them.

Model	Accuracy	F1-Score
BinaryRelevance	0.679214	0.78033
LabelPowerSet	0.495908	0.626289
MLKMM	0.274959	0.534687
Classifier Chain	0.376432	0.717443

It was very clear the Binary Relevance has the best results. But this is not the only reason.

Binary relevance is simple. Fast (linear complexity). It has some disadvantages as it ignores correlation between labels but it still manages to perform here as not all job functions should be correlated.

4 How can I extend the model to have better performance?

I don't think there is any extension the the model itself that can make a huge impact. the best way to extremely enhance our results is by using a Deep Learning model. Since we are dealing with texts. Using NLP methods and algorithms like words embedding/cnn. Will result in more understanding of the titles and the jobs.

5 Model Evaluation

Using accuracy and F1 score.

The F1 Score is the 2*((precision*recall)/(precision+recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between precision and recall.

```
Accuracy =  0.679214402618658
f1 Score is :  0.7803302225412778
```

6 Limitations

-Any non english words (eg:arabic) as we used stemming and also removed everything that is not character or number during data cleaning to get a better data.

-If job titles were heavily dependent on each other as the binary relevance doesn't really pay attention to that.

7 RESTful API

I used flask RESTful API as required. The restful script does the following.

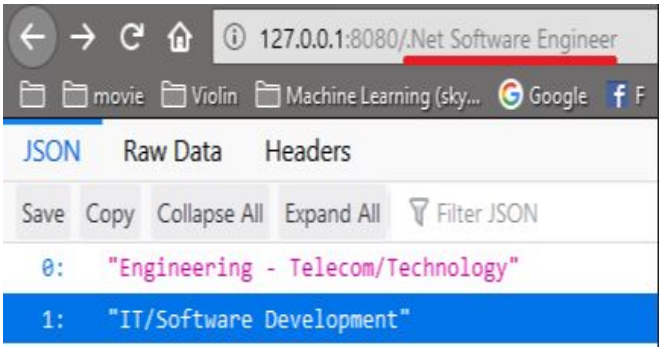
First it load vectorizer and trained model that were already saved and tested from the other script.

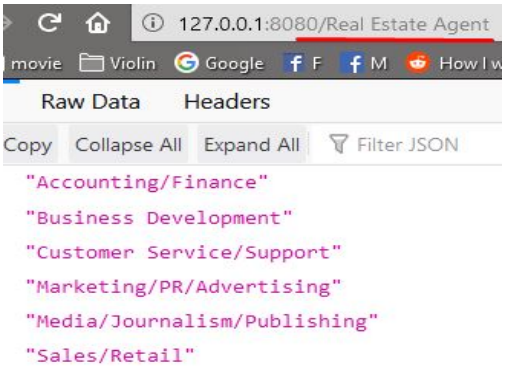
It contains a function that calls the dataframe and get our classes (all job functions).

In our main function (the routed one), it gets a title as html request written just after the url "/<title>".

I call the classes sort them. I then perform all the data cleaning on the title I made to our main data as they have to be the same (stemming, removing stop words, remove special chars).

Then I use the vectorizer on our title and then predict the outcome using our trained model. And return the classes that refer to the indices obtained from the prediction in json form.





8 References

https://journal.r-project.org/archive/2017/RJ-2017-012/RJ-2017-012.pdf

https://en.wikipedia.org/wiki/F1_score

https://link.springer.com/article/10.1007/s13748-012-0030-x

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

https://sebastianraschka.com/Articles/2014_naive_bayes_1.html