**Real/Fake Job Prediction**

**Shubham Kumar**

School of Information Studies,

Applied Data Science,

Syracuse University, Syracuse, NY, 13210

skumar11@syr.edu

**Abstract**

The process of applying for a job has almost completely moved to online systems. Online job portals attract many millions of users who are in search of a job. These portals allow employers to broadcast their postings to a very large audience of jobseekers. These millions of online interactions between jobseekers and employers lead to the possibility of Employment scam. The need to correctly identify whether instances of employment scam is paramount to maintain the reputation of online job portals and to prevent the loss of privacy for people who are the victims of employment scam. In this paper we use various text classification models to correctly identify fake and real jobs. The project comes close to replicating the results of a 2017 published paper on the same topic.

**1. Introduction**

Employment Scam has been defined as a “malicious behaviour aiming to inflict loss of privacy, economic damage or harm the reputation of the stakeholders of ATS” (Vidros et al., 2017) by manipulating the “job-advertisement publishing process” (Vidros et al., 2017). Scammers publish a job posting and pose as a real job on job-portals with the hope of luring candidates to apply for these roles. Scammers gain access to many resumes in this way and this is also a waste of time and effort for those who applied to these roles.

We use text classification techniques on text features such as job description, company profile, job requirements, and benefits to identify fake and real jobs. For human coders, it is not always clear why a certain job is fake and hence, Machine learning can help.

**2. Method**

**2.1. Data Description**

The Dataset we are using comes from a job portal called Workable. The dataset is present for public use under the name of Employment Scam Aegan Dataset (EMSCAD).

EMSCAD contains 17,014 authentic job postings and 866 fake job postings published between 2012 to 2014 (Vidros et al., 2017). The entries are manually annotated by human specialists at Workable. “The annotation process pertained to out-of-band quality assurance procedures. The criteria for the classification were based on client’s suspicious activity on the system, false contact or company information, candidate complaints and periodic meticulous analysis of the clientele” (Vidros et al., 2017).

Each record consists of 16 fields and one binary class field (real or fake). Among the sixteen fields are Binary, Nominal, String and HTML text fields as shown in Table 1.

Table 1: List of variables present in Dataset

|  |  |  |
| --- | --- | --- |
| **Type** | **Name** | **Description** |
| String | Title | Title of the Job |
| Location | Location of the Job |
| Department | Internal Department (eg. Marketing) |
| Salary range |  |
| HTML Fragments | Company Profile |  |
| Description |  |
| Requirements |  |
| Benefits |  |
| Binary | Telecommuting | True for Telecommuting positions |
| Company logo | True if logo present |
| Questions | True if screening questions presents |
| Fraudulent |  |
| Nominal | Employment Type | Full-time, contract, etc. |
| Required Experience | Entry level, etc. |
| Required Education | Master’s Degree, etc. |
| Industry | Information Technology, etc. |
| Function |  |

Due to the high imbalance in the dataset, we take a random sample of 450 job postings from each category. The original dataset also contains many empty fields, and we replace the empty values by empty characters before vectorization.

Due to the nature of human annotation, the dataset might have some jobs misclassified. We expect the number to be insignificant.

**2.2. Data Pre-processing**

**2.2.1. Stop word Removal**

Many Job postings contain similar words such as “customer”, “client”, “career”, “company”, etc. These words occur so frequently that they are going to be present in all our features while not adding value to the prediction accuracy. We remove these words from the four HTML fragments columns.

**2.2.2. Text cleaning**

The text columns (Company profile, Description, Requirements and Benefits) contain masked URLs, emails, and phone numbers. We compare model performance before and after pre-processing to remove non-alphabetic characters. The results are present in Table 3 for both the methods.

**2.2.3. Splitting the dataset**

Data is randomly divided into train and test datasets with ratio of 6 to 4. Training data includes 540 examples, while 360 records belong to the test set.

**2.3. Classification**

Before we can run any classification models, we do vectorization on the combined text columns. We combine the 4 columns separated by a space. For vectorization we experiment with four different types of vectorization techniques.

We are using three classification models here - Multinomial Naive Bayes, Random Forest and Support Vector Classifiers. For each of these models we experiment which vectorizer works best.

To do this we use three-fold cross-validation accuracy of these different vectorization methods as given in Table 2.

We choose the best vectorizer for each classifier and compare the default model accuracy.

Later on we also use grid search to find the best parameters that give the highest train accuracy. We compare the best parameter models with the default model on find which performs better on the test sample.

**3. Results**

The baseline majority vote model has 50 percent accuracy, 50 percent recall and 50 percent precision.

**3.1. Multinomial Naive Bayes (MNB)**

We use Python’s sklearn implementation of MNB. Table 2 shows the model’s 5-fold train accuracy using 4 different vectorizers. We see that the unigram Tf-Idf vectorizer has the highest accuracy of 86.5%.

Table 2: MNB Vectorizer accuracy

| **Vectorizer** | **5 fold Cross validation Accuracy** |
| --- | --- |
| Unigram Bool Vectorizer | 85.70% |
| Unigram Count Vectorizer | 85.30% |
| Gram12 Count Vectorizer | 83.10% |
| Unigram Tf-Idf Vectorizer | 86.50% |
| Cleaned - Unigram Bool Vectorizer | 85.40% |
| Cleaned - Unigram Count Vectorizer | 83.10% |
| Cleaned - Gram12 Count Vectorizer | 81.10% |
| Cleaned - Unigram Tf-Idf Vectorizer | 83.70% |

Using the unigram Tf-Idf vectorizer we run MNB models on raw data and cleaned data. Table 3 shows the test accuracy for these models. We see that the model with cleaned data and grid search parameters has the highest accuracy and highest recall.

Table 3: MNB model test accuracy RAW and Clean Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Vectorizer** | **Accuracy** | **Precision** | **Recall** |
| Default Model | 0.82 | 0.82 | 0.83 |
| Default Cleaned Model | 0.84 | 0.87 | 0.8 |
| Best Model GridSearch | 0.83 | 0.84 | 0.82 |
| Cleaned Best Model GridSearch | 0.84 | 0.85 | 0.83 |

**3.2. Support Vector Classifier (SVC)**

As shown in Table 4, the unigram Tf-Idf with cleaned data has the highest 5-fold train accuracy.

Table 4: SVC Vectorizer accuracy

| **Vectorizer** | **5 fold Cross validation**  **Accuracy** |
| --- | --- |
| Unigram Bool Vectorizer | 86.70% |
| Unigram Count Vectorizer | 82.70% |
| Gram12 Count Vectorizer | 84.40% |
| Unigram Tf-Idf Vectorizer | 87.40% |
| Cleaned - Unigram Bool Vectorizer | 86.50% |
| Cleaned - Unigram Count Vectorizer | 84.80% |
| Cleaned - Gram12 Count Vectorizer | 85.40% |
| Cleaned - Unigram Tf-Idf Vectorizer | 89.20% |

Using the unigram Tf-Idf vectorizer we compare the default model and the GridSearch best model. We run this analysis on the test data and do it for both clean and raw data. As shown in Table 5, the default clean text model performs best in terms of accuracy and recall.

Table 5: SVC test accuracy RAW and Clean Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Vectorizer** | **Accuracy** | **Precision** | **Recall** |
| Default Model | 0.86 | 0.87 | 0.86 |
| Default Cleaned  Model | 0.89 | 0.88 | 0.9 |
| Best Model  GridSearch | 0.83 | 0.84 | 0.82 |
| Cleaned Best Model  GridSearch | 0.88 | 0.88 | 0.89 |

**3.3. Random Forest Classifier**

We perform a similar analysis using a Random Forest classifier. The 5-fold train accuracy is highest for the Unigram Tf-Idf model using cleaned data (Table 6).

Table 6: Random Forest Vectorizer accuracy

| **Vectorizer** | **5 fold Cross Validation**  **Accuracy** |
| --- | --- |
| Unigram Bool Vectorizer | 86.80% |
| Unigram Count Vectorizer | 86.70% |
| Gram12 Count Vectorizer | 86.70% |
| Unigram Tf-Idf Vectorizer | 85.70% |
| Cleaned - Unigram Bool Vectorizer | 88% |
| Cleaned - Unigram Count Vectorizer | 86.50% |
| Cleaned - Gram12 Count Vectorizer | 86.50% |
| Cleaned - Unigram Tf-Idf Vectorizer | 88.70% |

Using the unigram Tf-Idf vectorizer, we get a highest accuracy of 88 percent and a highest recall of 91 percent (Table 7). This model is the default Random Forest Classifier in Python’s sklearn package.

Table 7: Random Forest test accuracy RAW and Clean Data

|  |  |  |  |
| --- | --- | --- | --- |
| **Vectorizer** | **Accuracy** | **Precision** | **Recall** |
| Default Model | 0.85 | 0.88 | 0.81 |
| Default Cleaned  Model | 0.88 | 0.87 | 0.9 |
| Best Model  GridSearch | 0.85 | 0.82 | 0.91 |
| Cleaned Best  Model  GridSearch | 0.85 | 0.84 | 0.88 |

**3.4. Bidirectional Encoder Representations from Transformers (BERT)**

BERT is a deep learning model which is pre-trained on NLP tasks.

In our experiments with BERT, we see a test accuracy of 81 percent and 79 percent recall. The confusion matrix of this model is shown in Table 11.

**4. Discussion**

**4.1. Confusion Matrix**

Confusion Matrix is another way to look at the errors the model is making and judge its performance.

Table 8 shows that the best MNB model makes 30 wrong predictions where it classifies a fake job as real. It also makes 26 errors where it classifies a real job as fake.

A further look at the prior type of error shows that the length of text columns (Company profile, Description, Requirements and Benefits) could add some value to our model. We also see luring phrases like “earn money by putting in a few hours” and some feature engineering to capture the presence of these might be helpful to further classify the fake jobs correctly.

Table 9 shows the confusion matrix for SVC. Both the errors are reduced by using this model. We can get recalls of 90% on par with the research paper. As seen with the previous model, luring phrases like “THE REWARDS ARE ENDLESS” are indicative of the job being fake and capturing these would improve the model.

Table 10 shows the confusion matrix for Random Forest Classifier. It has a lower error on predicting Real Jobs as fake.

Table 8: Confusion matrix - MNB + Cleaned Data + GridSearch Model

|  |  |  |
| --- | --- | --- |
|  | **Actual Real Job** | **Actual Fake Job** |
| **Predicted Real Job** | 153 | 30 |
| **Predicted Fake Job** | 26 | 151 |

Table 9: Confusion matrix - SVC + Cleaned Data + Default Model

|  |  |  |
| --- | --- | --- |
|  | **Actual Real Job** | **Actual Fake Job** |
| **Predicted Real Job** | 157 | 20 |
| **Predicted Fake Job** | 22 | 161 |

Table 10: Confusion matrix - Random Forest + Cleaned Data + Default Model

|  |  |  |
| --- | --- | --- |
|  | **Actual Real Job** | **Actual Fake Job** |
| **Predicted Real Job** | 154 | 25 |
| **Predicted Fake Job** | 19 | 162 |

Table 11: Confusion Matrix: BERT + Cleaned Model

|  |  |  |
| --- | --- | --- |
|  | **Actual Real Job** | **Actual Fake Job** |
| **Predicted Real Job** | 154 | 25 |
| **Predicted Fake Job** | 19 | 162 |

**5. Ethics**

As the jobs in the dataset are from the period 2012-2014, most of them should be expired by now. Any emails, URLs and phones, have been masked from the dataset.

The results of these classifiers should be used with caution as the prediction is not 100 percent correct and we are making some mistakes by tagging real job postings as fakes and failing to identify some fake job postings as real.

**6. Conclusion & Future scope**

In this paper we see that using Text mining methods we can identify fake jobs from real jobs with 89% accuracy. This task takes a lot of time for human annotators to do as they would have to go through each job and decide. Our experiments suggest that different classifiers perform better with different types of vectorizers.

For future work, we suggest using categorical features in complement to the text classification models. We saw that sometimes fake jobs have a very short text field (Company profile, Description, Requirements and Benefits) length. The presence or absence of luring phrases is also a valid avenue to explore. Using these features in complement to the text classification models can be helpful to improve the accuracy and develop more resilient models. In the future I would like to experiment with an ensemble approach to use text models and categorical features classifier models together.

**References**

[Vidros et al.2017] Sokratis Vidros, Constantinos Kolias, Georgios Kambouraki, & Leman Akoglu. 2017. Automatic detection of online recruitment frauds: Characteristics, methods, and a public dataset. *Future Internet*, *9*(1), 6