Real vs Fake Job Listing Classifier

Wylie Mickelson
mickelw@wwu.edu

Griffin Guyer
guyerg@wwu.edu

Abstract

This paper consists of the final project for our Natural Language Processing course. A dataset was found on the Kaggle website consisting of close to 18,000 unique data points containing various job listing information. We cleaned and tokenized the information to be used in a logistic regression model for classifying whether a job listing is real or fake. Later, we implemented a pretrained BERT model and fine-tuned it using the same dataset to expectantly get a better spread of results and a more accurate model.

1 Introduction

As job searching continues to move toward almost purely digital, it is becoming increasingly easier for companies to post fake job listings online. Some reports show that about 1/3 of job listings are fake. This can be a massive problem for job seekers. Applying for a fake listing wastes a job seeker's time, a valuable resource which could be better spent applying to actual companies. The time being taken to change one's resume and cover letter to fit a job, even sometimes going as far as interviewing for a position that is not truly being offered is an alarming waste of time. By modeling a classifier to indicate whether a job listing is real or fake, we will be able to assist applicants to spend the majority of their time applying to verified postings.

2 Related Work

The dataset we used is fairly popular online, so there were many resources we could pull from. Here is a description of two that we found using the same dataset:

The first paper[2] used five different models: Logistic Regression, Naive Bayes, Decisions Tree, Support Vector Machine, and K-Nearest Neighbors. They chose to focus only on the textual information from the dataset and split the data into training, tuning, and evaluation sets. Their results ended up being fairly good, with KNN being the lowest accuracy at 81.3% and Logistic Regression barely beating SVM for the top spot at 98.57%. Accuracy is the only metric they reported which can be often misleading in a model's effectiveness. This makes it difficult to really know how good these models were at catching fake jobs.

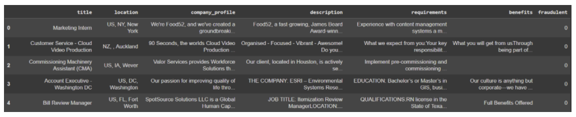
The second paper[3] used only two models: Logistic Regression and BERT, very similar to our work. They focused solely on three text-based columns of the dataset: job description, company profile, and job requirements. They split the data into only training and validation sets, no tuning. Their results were also pretty promising. Their Logistic Regression model had an accuracy of 99.45% and a mean absolute error of 3.27%. The BERT model had an accuracy of 99.81%. These were much better numbers than the previous work. This paper also showed their confusion matrices which allow us to see where their models make the most mistakes. Both models incorrectly labeled a fake job as real more than a real job as fake, which is the better of the two outcomes.

3 Experiments

Our approach to this task was to obtain a baseline classifier model using logistic regression paired with TF-IDF word embeddings. We would then go on to use a pre-trained BERT model and fine-tune it using the same dataset that was used for logistic regression. Once both models were obtained, an analysis and comparison of the two would be conducted.

3.1 Data Processing

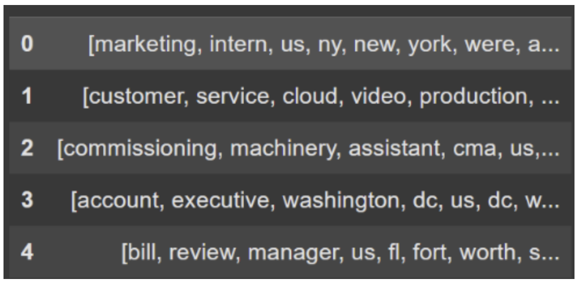
The dataset used for this experiment was found on Kaggle[1]. It contains data from 18,000 job descriptions in the format of a CSV file. A key note about this dataset is that it is highly imbalanced, and skewed in favor of real job listings. The ratio is about 95 percent real to 5 percent fake. This aspect will come into play later when we discuss the results of both models.



	title	location	company_profile	description	requirements	benefits	fraudulent
0	Marketing Intern	US, NY, New York	We're Food52, and we've created a groundbreaki...	Food52, a fast-growing, James Beard Award-winn...	Experience with content management systems a m...		0
1	Customer Service - Cloud Video Production	NZ, , Auckland	90 Seconds, the worlds Cloud Video Production ...	Organised - Focused - Vibrant - Awesome! Do you...	What we expect from you:Your key responsibilit...	What you will get from usThrough being part of...	0
2	Commissioning Machinery Assistant (CMA)	US, IA, Wever	Valor Services provides Workforce Solutions th...	Our client, located in Houston, is actively se...	Implement pre-commissioning and commissioning ...		0
3	Account Executive - Washington DC	US, DC, Washington	Our passion for improving quality of life thro...	THE COMPANY: ESRI – Environmental Systems Rese...	EDUCATION: Bachelor's or Master's in GIS, busi...	Our culture is anything but corporate—we have ...	0
4	Bill Review Manager	US, FL, Fort Worth	SpotSource Solutions LLC is a Global Human Cap...	JOB TITLE: Itemization Review ManagerLOCATION:...	QUALIFICATIONS:RN license in the State of Texa...	Full Benefits Offered	0

Figure 1: Initial Dataset

The columns containing textual data included a title, location, department, company profile, description, requirements, and benefits. The dataset contained other numerical information, such as salary and various binary data points. Since we are basing our models on natural language processing concepts, we elected to discard all non-textual information from the implementation process, excluding the identifier of whether the job was real or fake. Some of the columns were left empty, indicating there was no information for that particular data point. However, there was at least one textual column available for every data point, so a total of zero entries contained no text at all. To solve this issue, we elected to combine all of the textual data columns into one. Our resultant base dataset before modeling contained a single column with all combined text, and a binary identifier which indicated real or fake. It was then split into training and testing partitions with a 80 percent to 20 percent respective ratio.



0	[marketing, intern, us, ny, new, york, were, a...
1	[customer, service, cloud, video, production, ...
2	[commissioning, machinery, assistant, cma, us,...
3	[account, executive, washington, dc, us, dc, w...
4	[bill, review, manager, us, fl, fort, worth, s...

Figure 2: Modified Dataset

3.2 Logistic Regression Model

Going into this project, we needed to create some kind of baseline results. Logistic Regression was chosen because of our familiarity with it, and also since it's one of the easier ones to implement. The model that ended up being created was kept mostly simplified. Additionally, by making use of the helpful scikit-learn python packages, it was implemented fairly quickly. Using the scikit-learn functions, we calculated TF-IDF scores for both the training and testing partitions. Once they were calculated, we ran the scores through the respective training and prediction functions to produce our Logistic Regression model.

3.3 BERT Model

The idea for the second implementation was to use a transformer based model. Since this is a classification task, we only require the encoder portion of the transformer and can neglect the decoder half. The final decision was to use one of the most popular models, Google's BERT.

Our team had very little prior experience with the tools needed to implement machine learning models, including python and the various libraries required. This was a major hurdle going into the project since a lot of time was spent reading documentation and debugging errors to get a standard codebase working correctly. The code we used was being hosted on Google CoLab, and once we finally obtained a working setup, the training time for the BERT model exceeded 12 hours. This amount of time is longer than the total time a single instance of CoLab is able to run, so we needed to find a solution. Our initial idea was to use a smaller, more efficient subset of the BERT model, known as DistilBERT. It demonstrates similar accuracy and performance (while slightly lower) to the larger model, while being computationally faster. Unfortunately, this solution wasn't good enough. It only took the training time down to about 9 hours, which was still far too long. The second solution was to move our code and training to a local device, rather than rely on CoLab's much slower CPU. Our local CPU had the advantage of running indefinitely, and was also superior in processing power. However, we elected to use the GPU for training instead. Our GPU contained CUDA cores, which are perfect for optimizing tasks with parallel processing, such as the training and inference of a transformer model. This solution surpassed our expectations, lowering the training speed to only a half hour.

4 Results and Discussion

4.1 Evaluation Metrics

The evaluation metrics used to compare models contained common benchmarks for inference tasks, such as precision, recall, F-1 score, and accuracy. Due to the dataset imbalance in this case, some of the scores are more important to observe than others. Accuracy isn't as important as one might expect, since each model could theoretically predict every single test instance as a real listing and produce 95 percent overall accuracy. This is one of the problems with a skewed dataset. The main metric we'll take a focus on is the recall and precision. These give a better understanding of where the model is performing well and what could be improved. Additionally, confusion matrices will be provided for a visual representation.

4.2 Logistic Regression Model Results

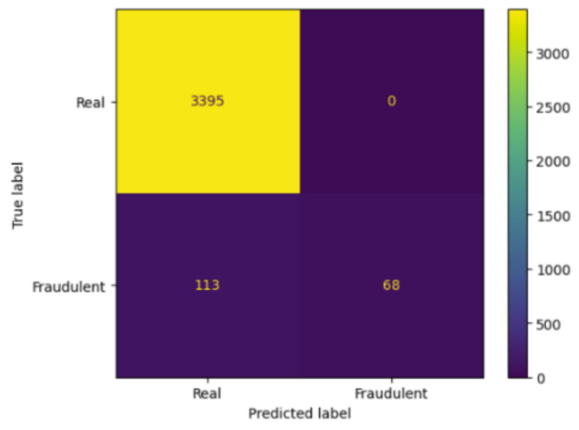


Figure 3: Logistic Regression Confusion Matrix

	Precision	Recall	F1-Score
Real	0.97	1.00	0.98
Fraudulent	1.00	0.38	0.55
	Accuracy	0.97	

Figure 4: Logistic Regression Metrics

The logistic regression model with TF-IDF gave a pretty decent baseline in terms of predictions. It had a perfect recall score identifying real job postings with the correct label, which is amazing. As for predicting fraudulent listings, it definitely fell short, with only 0.38 recall. This is most likely from the partition of the initial dataset. The logistic regression model was able to get incredibly good creating weights based on real job listings, since it had over 15 times more instances of that type compared to fraudulent ones. We believe this poor performance is because there wasn't enough fraudulent data points to train the weights correctly.

4.3 Transformer (BERT) Model Results

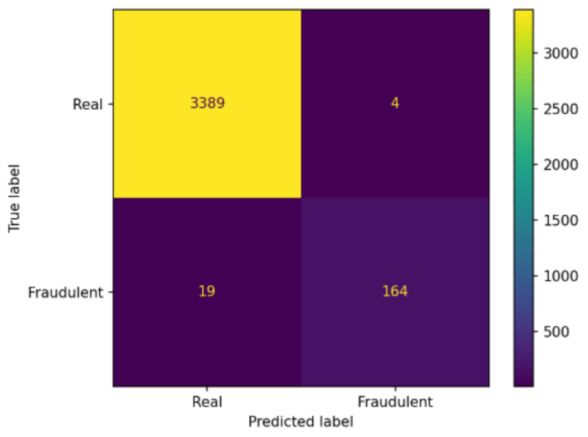


Figure 5: Transformer (BERT) Confusion Matrix

	Precision	Recall	F1-Score
Real	0.99	1.00	1.00
Fraudulent	0.96	0.89	0.92
	Accuracy	0.99	

Figure 6: Transformer (BERT) Metrics

The results of the fine-tuned transformer BERT model exceeded our expectations. It kept the high recall for real listings, with only 4 being mislabeled as fraudulent. The real improvement was with the fraudulent predictions, going from a 0.38 recall on the logistic regression model to a 0.89 with BERT. The overall accuracy of this model exceeded 99 percent, with a total of only 23/3,576 testing points being mislabeled. We believe the majority of the improvement between models was due to the use of the pre-trained BERT model. Since it was trained on a very large representation of the english language, it had a very good understanding of the intricacies and relationships between words and sentences. It was able to fill in the missing pieces of information from the lack of data given by fraudulent data points. The logistic regression model did not have this knowledge, and therefore performed worse.

5 Conclusion and Future Work

As stated earlier, our team was quite inexperienced in python and implementing machine learning algorithms. Through this project, we developed a good baseline knowledge and understanding of what goes into creating a real-life model and the issues that arise when attempting to do so. This will allow us to implement more advanced concepts in the future, since we will not be bothered by learning everything mostly from scratch. Despite the errors encountered, we are pleased with the end result of our experimentation with the two created models.

One extension of this project that we might like to pursue is modifying the original dataset to make it less imbalanced, and subsequently see how these changes might affect each model's results. There are a few ways to do this. The first, and fairly easy solution would be to use a subset of the real entries in the dataset. By keeping all of the fraudulent listings and using far less real ones, the overall proportions of the dataset can be more even. An issue with this approach is that depending on the final size of dataset used, it could end up being not enough information for the models to train in an optimal way. This might cause worse results. The second solution is a bit more difficult, and relies on data augmentation techniques to generate more instances of the fake data points. This approach would take the existing fake data and make augmented copies, which are similar but not identical to the original data. By evening out the imbalance this way, we would be able to keep the original size of our dataset, and only increase it. However, it's possible that we don't have enough fake data points to create an accurate representation on the same scale as our real data points. By creating too many new instances, we risk making a flawed dataset that doesn't reflect the structure of the original entries, which fail to accurately represent the fake listings as they would appear in the real world.

6 Code Base

Our codebase contains the initial Google CoLab file and a local python script. The CoLab contains all code relating to the data processing and creation of the logistic regression model, while the local script has all the contents of the BERT transformer model. There is also a saved csv file containing only the combined text and label columns of our dataset, which was primarily used for the local script. Github.

References

[1] Shivam Bansal. "Real / Fake Job Posting Prediction Dataset". In: (). URL: https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction/data.

[2] et al Devansh Petal. "REAL/FAKE JOBS CLASSIFICATION (EDA)". In: *IRJMETS* (). URL: https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2024/50580/final/fin_irjmets1711647736.pdf.

[3] Ronit Munshi. "Predicting Fake Job Listings from Real Ones using Machine Learning Models". In: (). URL: https://independent-project-mentorship.netlify.app/assets/pdfs/e41bc1e2b1380906e267fd864dea050181794d8f.pdf.