Resume Classifier Project Report

Overview:

Implemented a Naive Bayes model in Python learns from a dataset of resumes for various occupations to classify the jobs of new resumes while also displaying the performance metrics to analyze the accuracy of the model for further improvement.

Implementation:

In this project, I utilized the following Python libraries: nltk, pdfplumber, pandas, and sklearn. A Naive Bayes model will be used for training. This model will be trained on variables X and y, which will be defined later.

Step 1:
Preprocess the training resumes. To do this, I tokenized the text, then applied lemmatization and removed stopwords. Lemmatization is reducing a word to its base form, e.g. lemmatization of running, runs, and ran will be run. Stopwords are filler words such as ‘the’ and ‘a’.

Step 2:
Convert the training resumes into a Bag-of-Words model. Essentially, for each resume, a vector is created. Each element of the vector represents the count of a word in the resume. This will be our X variable for model training (feature).

Step 3:
Each resume in the training dataset corresponds to a label. We will convert these labels into a pandas series, which is an array where each entry has a label (just indices in my case), and each value is a job role. This will be our y variable for model training (class).

Step 4:
Split the resumes into two parts: training and testing. The test size will be 0.3, which means that 30% will be for testing and the remaining 70% will be for training. Now, we can initialize and train the Naive Bayes model.

Step 5:
Predict the job roles of the test resumes and output the predictions along with the accuracy statistics of the model. The accuracy statistics consist of precision, recall, f1-score, and support along with accuracy, macro average, and weighted average.
Precision: How many of the things you guessed as positive were actually positive.
Recall: How many actual positives you guessed correctly.

F1-Score: Balance between precision and recall. If both precision and recall are high, the f1-score will also be high (this is what we want). Formula: 2 x precision x recall / (precision + recall).
Support: How many real examples belong to each class (job role); i.e., the true count of each resume type.
Accuracy: How often the model is right.
Macro Average: The average score of each job type is weighted equally.
Weighted Average: The average score of each class when giving larger classes more weight; i.e. job roles that appear in the dataset more frequently will be weighted more.

Results:
My training dataset consisted of four job roles with three resumes for each job type. My testing dataset consisted of one resume for each of the four jobs from the training resumes as well as two more resumes that were of different job roles that were not in my training set.
The performance metrics of the model did well, and each test resume that was of a job role that appeared in my training set was predicted correctly. However, the two test resumes that were of different job types not included in the training set were labeled with job roles that did not seem to be closest to the correct job type by human judgement.

Contributions:

Tiffany Ho:

Contributions:
I worked alone, so I implemented all the NLP tasks as well as data gathering and research. I implemented the Naive Bayes model, converted the training dataset into a Bag-of-Words model, and preprocessed the data. Preprocessing consisted of lemmatization, removal of stopwords, and tokenization. I then implemented model training, such as splitting the dataset and initializing and training the model. Lastly, I implemented model prediction of the test resumes as well as generating model performance metrics.

Lessons Learned:
To improve my model, I would make my training dataset more large and diverse. I would do this by adding more job types and including more resumes for each job type.
While I did improve this slightly from my earlier prototype, as I started out with less job roles and resumes in each job type and added on to it to get to the final prototype for this project, it is still rather small, and would benefit greatly with more expansion.

Additionally, I would add TF-IDF, so that certain more significant keywords that may occur more often in a certain job type would have more precedence over less significant words.

Self Score:

Tiffany Ho:

75/80 points - significant exploration beyond baseline -
I encountered the issue of not having a large or diverse enough dataset, and fixed this partially by expanding my dataset both in size and diversity of job roles.
If I had done this even more, the model could have performed significantly better with a larger variety of job types.

10/10 points - highlighted complexity -
When gathering data, for each job type, I would choose resumes that were slightly different in roles while still being the same job so that many varieties of each job could be predicted during testing.
Additionally, for model training, I utilized Bag-of-Words model along with Naive Bayes to try and maximize model performance.

10/10 points - discussion of lessons learned and potential improvements -
As discussed previously, I realized that in order to eliminate inaccuracies of job types not occurring in my training set, I would need to add more job roles in my training set as well as adding more resumes to each job type. This will significantly improve the information that the model has access to and significantly improve performance.