Tentative Idea of CV Classification

Ponkoj Chandra Shill

An automated system on CV classification is a huge task to perform, as there are several data difficulties as well. Depending on the skills may not be efficient to find the candidates for a position there can be some other data dependencies that should be kept in mind to predict the targeted data. However, here I have demonstrated a tentative procedure that can be followed to solve this issue.

CV Data:

First of all there is a quiet difference between CV and articles or newspapers. The main difference in CV documents is in the structure. The CV format does not follow the same structure, hence text data is extracted depending on several common fields that are concerned. For instance, the candidate names, their contact information, skill set, work experiences, educational background, these are the information the recruiter looks for in each CV rather than the CV structure.

To solve this, there are some text parser libraries available in python that can be used to get structured data from a resume. For example, there are several libraries; PyPDF2 can extract data as text from pdf files. Moreover, 'pyresparser' has some features that is an efficient one which generates the information based on different fields from a resume. However, we can build our own parser based on specific requirements using libraries that are incorporated with natural language processing.

Important and irrelevant Information for Classification:

One of the most important parts is the data relevancy. The important and least relevant information or data totally depends on the requirements. For instance, if I need to classify resumes for python developers or java developers etc. in that case, the skill sets of each candidate would be prioritized as relevant information according to problem context. Moreover, the other personal information won't impact on the classification problem, hence those would be the least relevant data that can be ignored. However, incorporating the distinction between various levels of relevance into the model depends on the problem totally. Therefore, there should be a hybrid model inside the main model to cluster and handle the noise data to extract the important information only.

Features and Target:

Feature engineering and feature selection is the important part for all machine learning or data science projects. About the targeted feature for CV classification, I would like to take observations like the set of skills as binary features from the skill section, not the whole cv. Because, when you are going to classify a CV whether it is python developer or java developer, you have to check the skill sets. In that case, in the feature selection procedure the independent features would be the skills and the dependent / targeted feature would be the python developer, java developer, React Developer etc.

Model Type:

Once done with the preprocessing and feature selection and engineering procedures, the modeling part comes. Hence, I would develop a hybrid model for the whole project. In the first layer, a clustering algorithm will cluster the resumes in different clusters like IT, business,HR, pharmacy , law etc based on overall information along with the experiences, designations, education etc. Another layer for text analysis where people use a whole sentence instead of bullet points, we can perform text classification using popular methods like BERT. or LSTM with embedding layers like Glove Embedding method. The main layer model will classify the CV on specific skills like python developer or java developer etc. There might be a few more layers of model included based on requirements and purpose. And the core layer of the classification layer on several skills will be the same as the steps I will below. All the layers of different models will be introduced to help the core layer or the skill classification model for better performance and to achieve robustness along with the variety of features (experience, education, certification , soft skills etc.) and edge cases(long text, indirect skills presentation etc.).

Here I am sharing a tentative or rough plan for CV classification using the set of skills of a CV. Henceforth, this tentative plan can be considered as a benchmark for a CV classification model so that we can modify and improve our whole hybrid model compared with this one. In the benchmark model, I would like to use only the technical skill sets to classify the CVs in the first place.

Steps for Main layer of the Classification on several skill sets:

- Get resume in PDF format :
  A certain amount of labeled resumes need to be collected for the model training and evaluation process.
- Using available parser to get data from resume. For example ‘pyresparser’ library is a simple resume parser library used for extracting information from resumes including the features-
  - Name : Ponkoj Chandra Shill
  - Contact: +8801521333203
  - Email : csponkoj@gmail.com
  - Skills : [Python, Jupyter Notebook, Pandas, …]
  - Experiences: [...]
  - Certificates: [...]
  - Education:[...]
- After that, get the Skills key with the list of values from the whole data for further processing
  - For Example : Skills : [Python, Jupyter Notebook, Pandas, …]
- Do some processing in text data (skills)
  - Remove stop words : using mostly used text processing library, NLTK
  - Processing with Regular Expression : to remove special characters, URLs, white spaces, single characters, digits, uppercase letters, substituting multiple spaces with single space
- Do lemmatization using Wordnet Lemmatizer on the list of processed skills, to get the base words

- Use FuzzyWuzzy for text matching, to get only the valid set of skills, which will be trained
- Prepare the data:
    - Use email as index
    - Make set of skills as binary features

Email	Python	Java	C	C++	Tensorflow	Keras	Pandas	Label
abc@gmail.com	1	0	0	0	1	1	1	Data Scientist

Figure : Features with labels

- Generate the Dataset
    - Get each data points for a single resume with a list of binary features
    - Merge all the data points into one dataset
- Do some Feature Engineering on the dataset
- Split the dataset into train and evaluation set
- Select Classification model for training
- Use Bayesian Optimization for hyperparameters tuning
- Train the dataset with classification model
- Evaluate the model with evaluation set
- Keep track of misclassified CVs, for further improvement of model
- Iteratively improve the model

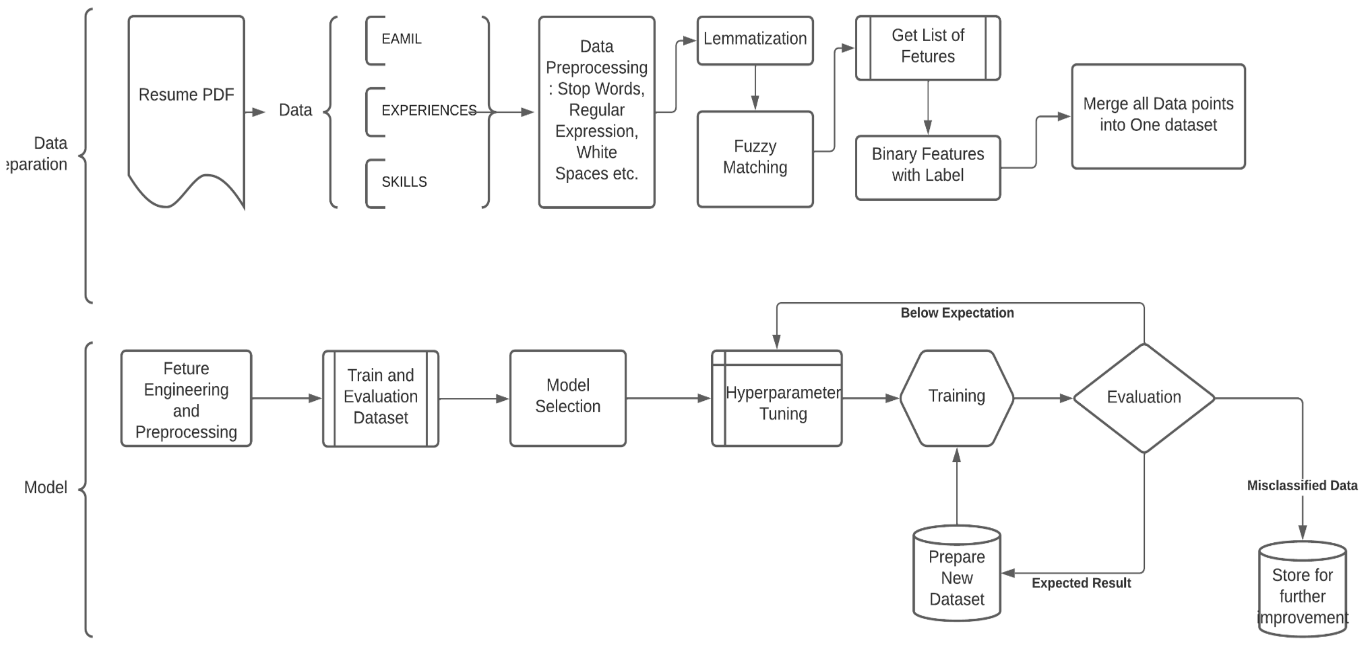

Data
eparation
Resume PDF
Data
EAMIL
EXPERIENCES
SKILLS
Data Preprocessing : Stop Words, Regular Expression, White Spaces etc.
Lemmatization
Fuzzy Matching
Get List of Fetures
Binary Features with Label
Merge all Data points into One dataset
Model
Feture Engineering and Preprocessing
Train and Evaluation Dataset
Model Selection
Hyperparameter Tuning
Training
Evaluation
Below Expectation
Misclassified Data
Prepare New Dataset
Expected Result
Store for further improvement

Figure : Tentative model plan

Common Issues:

There are challenges in data processing steps, using a prebuilt parser to parse info from a CV would not be 100% accurate for all CVs. Moreover, in the near or future we need to build our own parser to parse info from cv based on our model requirements. There are several ways to generate dataset for model input. Like binary features and categorical features. We need to try our model both ways for research purposes to get a better model. But I would try binary feature input first.

There are several challenges at the production level that should be kept in mind such as data distribution changes, model Pipeline health issues, causes of extreme events (Outliers) etc. There can be low quality data annotation as humans annotate the data, they don't perform like machines hence it's normal to have the biased, noisy data in the dataset. To overcome the bias, data balancing can be an efficient methodology where it is ensured that the dataset is well balanced based on labels. Furthermore, oversampling methods for data balancing such as random oversampling, SMOTE can be efficient to handle this issue. However, there are some boosting algorithms that are considered robust models such as AdaBoost, Catboost, LIghtGBM etc. can also be followed in this regard.

Evaluation:

To evaluate potential candidates on this model, I would like to get all CVs in one place and classify all the CVs by chunk to get the potential CVs that belong to the required class. There are several evaluation matrices for monitoring the model performance. For instance, model accuracy level defines the model performance, the efficiency level basically. Moreover, I would like to analyze the misclassified CVs to track the performance of the model. Analyze the reason why those CVs are not classified correctly, is there any noise or bad data or actual model's issue? Moreover, visualizing the performance on live data, handling the new data in the model and checking the accuracy constantly is the part of monitoring model performance.

Model Improvement:

Over time the model might deteriorate due to new types of data, newer skills set etc. To overcome this, I would prefer an iterative cycle to retrain the best model with newer datasets or new features. To make it more efficient, an automated retraining cycle could be introduced here that can process the new data and retrain the model with transfer learning. Additionally, the same hybrid model pipeline would be used to classify new unlabeled data. Then the classified CV data will be used to retrain the model in the same pipeline. Moreover, keeping track and analyzing new skills or information that were not used in the previous model could be used as a feature for further improvement. These steps would be done in an automated manner so that the model can improve by itself iteratively.

- how would you make sure that your model is not deteriorating over time (e.g. when new, earlier unknown skills are reported in the incoming CVs)?

- Over time the model might deteriorate due to new types of data, newer skills set etc. To overcome this, I would prefer an iterative cycle to retrain the best model with newer datasets or new features. To make it more efficient, an automated retraining cycle could be introduced here that can process the new data and retrain the model with transfer learning

- how would you make sure that your model is not biased because of the subjective decisions or errors made by human annotators?
- Data balancing(each category have smae amount of ata point) , oversampling(random , SMOTE) , robust models like boosting algorithms(AdaBoost, Catboost, LIghtGBM etc)

  - There can be low quality data annotation as humans annottates the data, they don't perform like machines hence it's normal to have the biased, noisy data in the dataset. To overcome the bias, data balancing can be an efficient methodology where it is ensured that the dataset is well balanced based on labels. Furthermore, oversampling methods for data balancing such as random oversampling, SMOTE can be efficient to handle this issue. However, there are some boosting algorithms that are considered robust models such as AdaBoost, Catboost, LIghtGBM etc. can also be followed in this regard.

- how would you roll out your model to CVs for other sets of skills (e.g. HR positions instead of IT positions)

  - A Hybrid model would be implemented here. In the first layer, a clustering algorithm will cluster the CVs like HR, Business, IT based on overall information along with the skills, designations, education etc.

  .

- how would you monitor model performance over time?

  - There are several evaluation matrices for monitoring the model performance. For instance, model accuracy level defines the model performance, the efficiency level basically. Moreover, I would like to analyze the misclassified CVs to track the performance of the model. Analyze the reason why those CVs are not classified correctly, is there any noise or bad data or actual model's issue? There are several challenges at the production level that should be kept in mind such as data distribution changes, model Pipeline health issues, causes of extreme events (Outliers) etc . Moreover, visualizing the performance on live data, handling the new data in the model and checking the accuracy constantly is the part of monitoring model performance.

- how would you use the information contained in unlabeled CVs to retrain and improve the model?

- Same hybrid model pipeline would be used to classify new unlabeled data. Then the classified CV data will be used to retrain the model in the same pipeline. Moreover, keeping track and analyzing new skills or information that were not used in the previous model could be used as a feature for further improvement. These steps would be done in an automated manner so that the model can improve by itself iteratively.

-
-
-
-
-
- **How are CV documents different from other text data (e.g. newspaper articles)?**

- The main difference in CV documents is in the structure basically. All the CV format does not follow the same structure, hence text data is extracted depending on several common fields that are concerned. For instance, the candidate names, their contact information, skill set, work experiences, educational background, these are the information the recruiter looks for in each CV rather than the CV structure. Hence, some libraries are there with these common fields as features to extract the data which can be efficient in this regard.

- **How would you leverage structural information contained in a CV?**

  - There are some text parser libraries available in python that can be used to get structured data from a resume. For example, there are several libraries; PyPDF2 can extract data as text from pdf files. Moreover, 'pyresparser' has some features that is an efficient one which generates the information based on different fields from a resume. However, we can build our own parser based on specific requirements using libraries that are incorporated with natural language processing.

- **What kind of information would you find most relevant and least relevant?**

- The important and least relevant information or data totally depends on the requirements. For instance, if I need to classify resumes for python developers or java developers etc. in that case, the skill sets of each candidate would be prioritized as relevant information according to problem context. Moreover, the other personal information won't impact on the classification problem, hence those would be the least relevant data that can be ignored.
-

- **How would you incorporate the distinction between various levels of relevance into the model?**

  - That depends on the problem totally.

- **How would you verify your hypotheses about what is important information and what is a noise?**
    - There is high chance

- **what kind of input observations would your model be trained on (e.g. complete CV, individual sections) and why? What would be your target/label?**
    - I would like to take observations like the set of skills as binary features from the skill section not the whole cv. Because, when you are going to classify a cv whether it is python developer or java developer, you have to check the skill sets. In that case, in the feature selection procedure the independent features would be the skills and the dependent / targeted feature would be the designations (python developer, java developer, React Developer etc.)

- **Would you train just one model or individual models for different skills and why?**

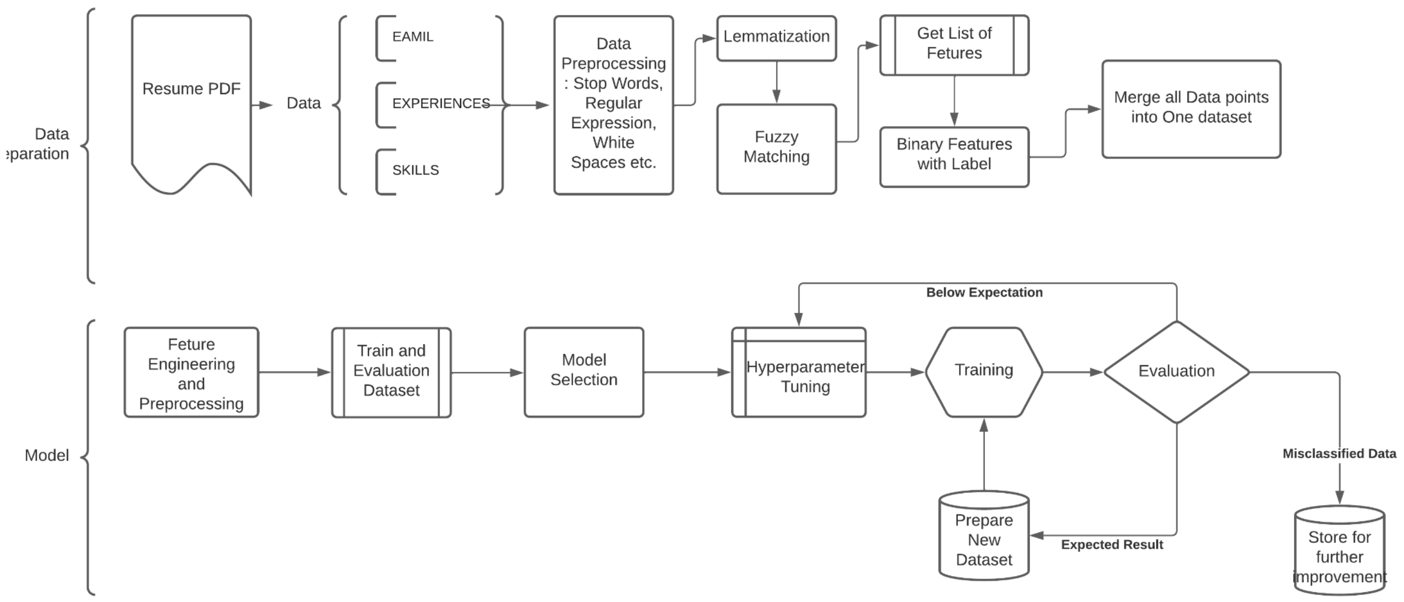
  - I would develop a hybrid model for different skills. For example, the 1st layer model will cluster the resumes in different clusters like computer science, business, pharmacy , law etc. Then the 2nd layer model will classify the cv on specific skills like python developer or java developer etc. There might be a few more layers of models included based on requirements and purpose.

- **what challenges do you expect in data preprocessing step (i.e. turning CV document into a model input)**
    - Using a prebuilt parser to parse info from cv would not be 100% accurate for all CVs.
    - Moreover, near or future we need to build our own parser to parse info from cv based on our model requirements
    - There are several ways to generate dataset for model input. Like binary features and categorical features. We need to try our model both ways for research purposes to get a better model. But I would try binary feature input first.
- **How would you evaluate potential candidates in order to decide on the best model?**
    -

Tentative Procedure:

- Get resume in PDF format :
  From the dataset each resume would be parsed for data extraction and the data will be merged together as a dataframe for further processing.

- Using available parser to get data from resume. For example ‘pyresparser’ library is a simple resume parser library used for extracting information from resumes including the features-
  - Name : Ponkoj Chandra Shill
  - Contact: +8801521333203
  - Email : csponkoj@gmail.com
  - Skills : [Python, Jupyter Notebook, Pandas, …]
  - Experiences: [...]
  - Certificates: [...]
  - Education:[...]
- After that, get the Skills key with the list of values from the whole data for further processing
- Do some processing in text data (skills)
  - Remove stop words : using mostly used text processing library, NLTK
  - Processing with Regular Expression : to remove special characters, URLs, white spaces, single characters, digits, uppercase letters, substituting multiple spaces with single space
- Do lemmatization using Wordnet Lemmatizer on the list of processed skills, to get the base words
- Use FuzzyWuzzy for text matching, to get only the valid set of skills, which will be trained
- Prepare the data
  - Use email as index
  - Make set of skills as binary features
- Generate the Dataset
  - Get each data points for single resume with a list of binary features
  - Merge all the data points into one dataset
- Do some Feature Engineering on the dataset
- Split the dataset into train and evaluation set
- Select Classification model for training
- Use Bayesian Optimization for hyperparameters tuning
- Train the dataset with classification model
- Evaluate the model with evaluation set
- Iteratively improve the model


Data
eparation
Resume PDF
Data
EAMIL
EXPERIENCES
SKILLS
Data Preprocessing : Stop Words, Regular Expression, White Spaces etc.
Lemmatization
Fuzzy Matching
Get List of Fetures
Binary Features with Label
Merge all Data points into One dataset
Model
Feture Engineering and Preprocessing
Train and Evaluation Dataset
Model Selection
Hyperparameter Tuning
Training
Evaluation
Below Expectation
Misclassified Data
Prepare New Dataset
Expected Result
Store for further improvement