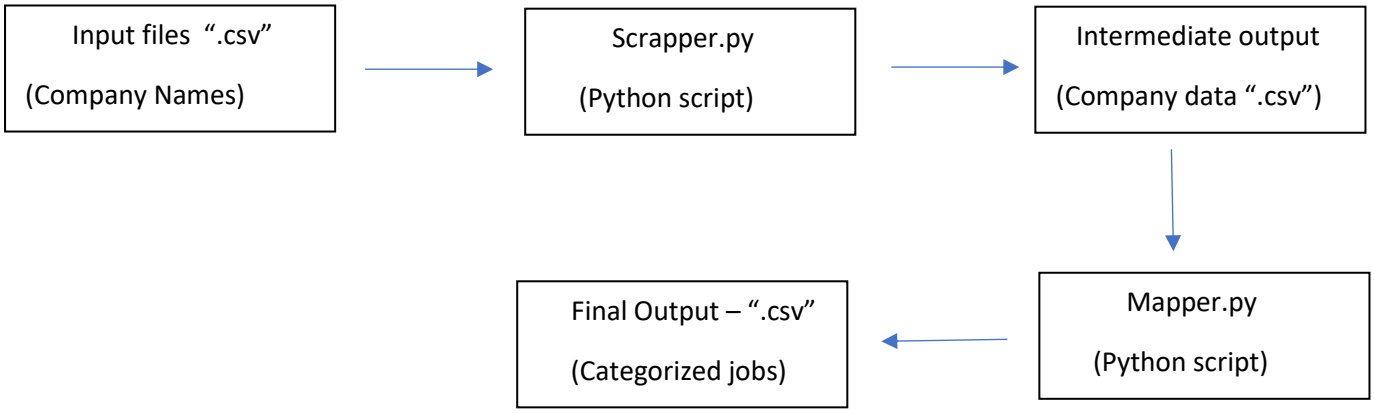
Report

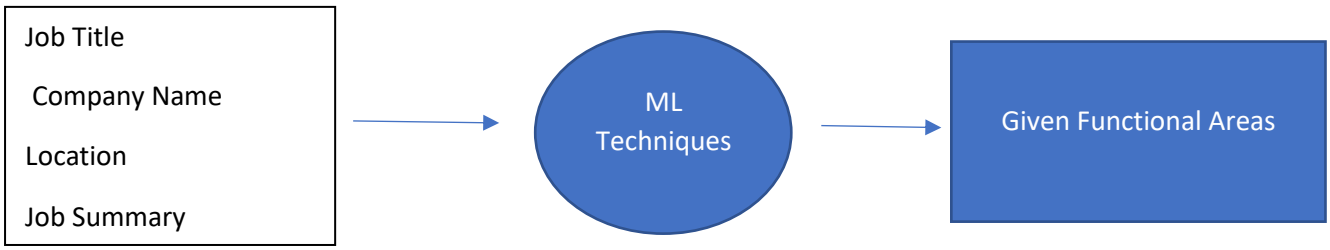
Project Basic workflow:



Problem statement formation:

1. Given list of features like:
    a. Job Title
    b. Company Name
    c. Location
    d. Job Summary

It is required to use them to categorize each job into six given functional areas like "Sales and Marketing", "Engineering, Research and Development", "Customer Services", Business Operations", Leadership", "Other"



Initial Questions to be asked :

1. Can this problem be solved by labelled data making it supervised?
2. If I do not have access to labelled data, Can it be solved using Unsupervised Learning method called Clustering?
3. How can I generate or extract labelled data?
4. Can I apply techniques like PCA() or LSA(latent semantic analysis) or LDA(latent dirichllet Allocation) to extract in-built structure or cluster of jobs?
5. If I generate Labelled data, will I be sure that I do not induce bias while I label the data and Do I have the required domain expertise to classify each job as it should be?

Thought Process in reply to these above-mentioned questions:

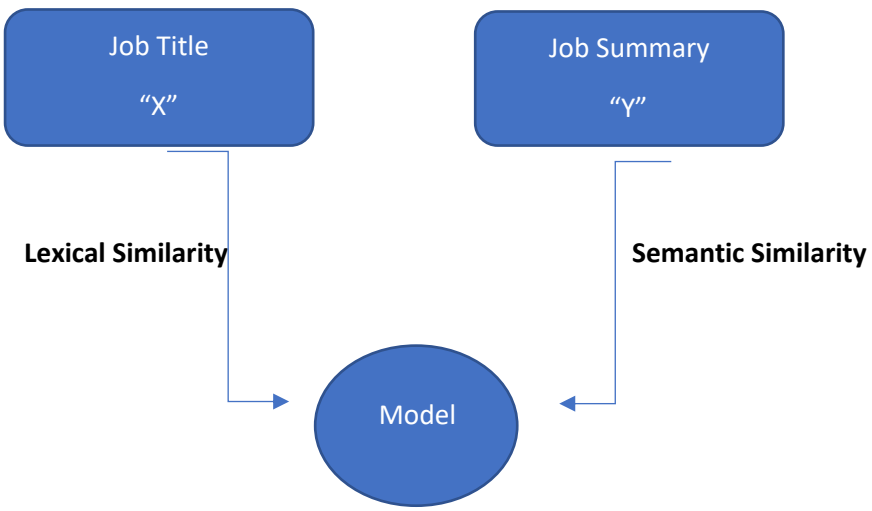
1. I could not find labelled data from other job portals, and I am not sure I would not induce bias while labelling. Hence supervised learning is out of equation.
2. I tried to cluster the job titles and job summary features using K-Means and found many overlapping content in multiple clusters and the cluster centres were not explanatory.
3. Using feature extraction techniques like PCA() would give me uninterpretable features which I cannot use to classify and cross-verify the classifications.
4. Also the clusters generated by the Unsupervised techniques need not just converge to the given six job functional areas in the problem statement.

Narrowing the solution:

1. I found that using Pre-trained word embedded models are a possibility given that I do proper text pre-processing.
2. I found few NLP models that had large vocabulary encoded in terms of word vectors that would solve this problem of mapping by using the similarity measure(cosine distance).
3. Now I need to narrow down on the type of models suitable for this problem. Spacy's word embedded model of large version had word embedded in high dimensions and vocabulary sizes were in millions.
4. On the other hand, Standford's Glove Model was called Universal Sentence Encoder used to handle sentences similarity.

Handling 2 types of Similarity measure for the respective features:

1. Lexical Similarity – This measures word-wise similarity and gives the similarity score based individual word distances in the embedded dimensions. This is done by the Spacy's Model and this can be used to handle the Job Title feature.
2. Semantic Similarity - This takes multiple words into account to provide a similarity score based on semantics. This is handled by the Glove model that processes the Job summary feature in this problem.
3. Now I need to take into account these 2 features and frame a scoring model that gives out a score and it could be used to map to the right functional area.



Deciding the feature coefficients:

Question: How do I decide the feature weights for the features X and Y ???

Initial Solution: Decide/ assume a set of feature weights and manually check the categorized jobs.

Cons: The number of jobs is huge to perform this checking so not possible.

The question that led to choosing the right metric for this optimization problem:

How similar the jobs must be to each other that are categorized under same functional area to verify our results???

So the metric born out of this question is the **"Intra-Cluster tightness**", this measures how similar the jobs are to each other given they belong to the same cluster.

So we just need to calculate this metric and find the pair of feature weights that maximizes this intra-cluster metric score.

Customer services cluster

Job-1

Job-2

……

Job-n

Leadership cluster

Job-1

Job-2

……

Job-n

…….(similarly for other clusters)

Here in each cluster we check how similar are the job to one another making it n(n-1) comparisons per cluster. This gives us a summed score and we need to take the summed scores of all the clusters and average them.

Intra-Cluster score = sum(each cluster's job tightness) / number of clusters (5)

I did not include the "**Other**" category since the jobs in this cluster have almost no relevancy.

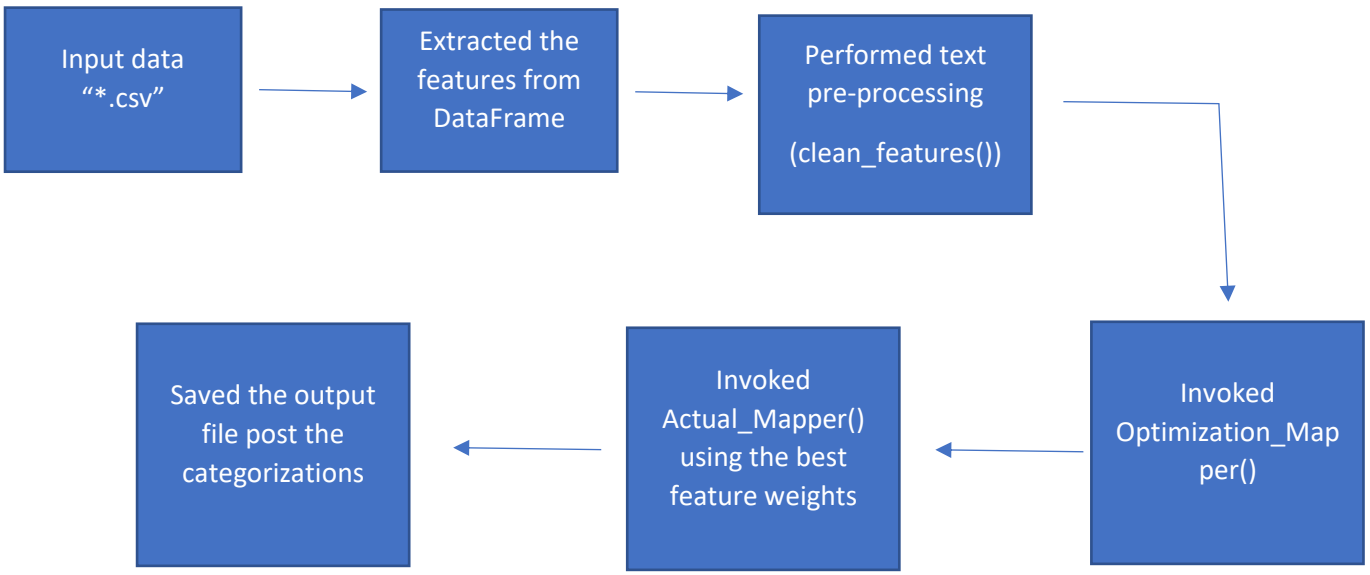
Trick used to classify jobs as "Other" category:

If all the 5 clusters give a similarity score below 50%, then it should be assumed that none of them are confident in having that job in their cluster meaning it belongs to the "Other" category.

Program walk-through:

1. **Scrapper.py**
    - I have read the input file that consists of company names and made it a python list.
    - Looping this list, I scrapped the jobs for each company in the list from indeed.com website.
    - I added a pause factor using **time.sleep()** function so that the site does not block my ip address.
    - Finally stored the results in csv files for next mapping step.

2. **Mapper.py**
    - Starting from Main function, using the glob package I extract all the stored "*.csv" files from the directory and concatenate them into one large dataframe.
    - From this I take the job title and job summary features and perform initial text pre-processing like:
        a. Stop words removal
        b. Punctuation removal
        c. Normalization through lemmatization
        d. Regular expression processing
    - Defining the list of feature weights to be tried.
    - Call the **optimization_mapper()** with parameters – **feature_weight**, **job titles cleaned and job summary cleaned version**
    - **optimization_mapper()** function tokenizes the input features and calculated the similarity scores using the pre-trained models and type of feature weights selected.
    - **optimization_mapper()** invokes the **intra_cluster_tightness()** function that calculates the metric scores for each pair of feature weights.
    - This calculation is done for all set of feature weights and the one that has the maximum relevancy score is selected as the final optimal feature weight for the model.
    - Now using the best feature weights, the **Actual_Mapper()** is called and the categorization happens for the jobs for each company.
    - Finally the results are stored in a dataframe which is converted to a "output.csv" file that contains the final output in the required format.



Finally How would I solve the same problem if I had access to labelled data:

- First I would try K-nearest neighbour method, using this I would get the relevancy scores of top K jobs for a given job and I would take the mode of the resulting categories from the label column.
- I would always choose the value of K as odd since it would always have an edge when it comes to choosing the mode which is a kind of voting scheme so choosing K as odd would lead to majority.
- Also we could try to extract features using LSA() or PCA() and see if we can with fewer dimension produce a good classifier.
- Further analysis depends on experimentation subject to the data at hand.