# ResumeRise - A Resume Classifier & Summarizer

1st Manish Shetty M 01FB16ECS192
*CSE dept, 5th Semester*
*PES University*
Bangalore, India
mmshetty.98@gmail.com

2nd Megha K 01FB16ECS203
*CSE dept, 5th Semester*
*PES University*
Bangalore, India
meghakalal09@gmail.com

3rd Meghana 01FB16ECS204
*CSE dept, 5th Semester*
*PES University*
Bangalore, India
meghana.holla@gmail.com

***Abstract*— ResumeRise is a categorization and summarization tool for resumes. Given a resume in PDF format, our model extracts any form of text from the pdf using Optical Character Recognition (OCR). This text was cleaned by by transformation to a better encoding scheme, removal of punctuation marks, stop words, and everything was reduced to lower case to avoid redundancy. Following the rigorous cleaning, the text was passed to our model for classification, which accurately predicts the category of resume. The overall output of our model would be the class of the input resume along with a summarized version of the resume which pulls out only the significant features of the resume.**
**For the process of classification, a range of models were tried out ranging from Multinomial Naive Bayes, Support Vector Machines, Logistic Regression. After repeated cleaning and training, our model was seen to perform fairly well with an overall accuracy of 70% using Support Vector Classifier.**

## I. INTRODUCTION

Resume is a pivotal part of the today's working world. Institutions concerning Graduate/Post-Graduate studies churn out thousands of students ready to be a part of global workforce and with them, come their resumes. Statistics say that a typical medium-large company gets around 100-200 job applicants per position. Generally, companies receive 50 to 75 resumes for each skilled position and as high as 300 for customer-service and entry-level positions. The number of resumes posted per week on employment websites like Monster, Indeed etc can go up to a whopping 27,000!!
The least that can be concluded from these statistics is that the number of resumes that a company receives is not less. From a recruiter's standpoint, going through each of these resumes, filtering them, bringing them down to a few and going through them again can prove to be time consuming and exhausting. Automating this tedious task of sorting out the resumes and extracting only the useful information would prove to be a boon to the recruiters. We here, propose an end-to-end model, that would do the job for the recruiters.

## II. LITERATURE REPORT REVIEW

### A. Initial Procedure

From the literature review procedure , we clearly did identify that not many attempts have been made to tap into the vast data that is collected all over the world in the form of resumes and how it can prove to be a vital place to apply automation into. Thus giving us a chance to work on a fresh problem in a more individually designed fashion specifically for the data-set chosen rather than following the typical procedure and optimizing on it.

The project on it's initial course began with a lot of things that were flawed but not visible until experimented with. The initial steps taken based on previous works was to use an annotation tool to annotate the important parts of a resume. The annotated resumes were then thought to be used as training data for a model.
The workflow included making use of Spacy , a module in python that has inbuilt pipelines for NLP and summarization. But on the course of this trial and error experiment , we realized that the model was severely over-fitting and was of no use to the goals we had for the project.
It had a number of flaws:

1) Data always had to be annotated and inflexible in a Json format
2) The initial results of the model were above 95% accurate showing clear signs of a void in it's generalization.
3) The classification problem , which was our main goal remained unsolved.

The dataset that we had chosen during the time of literature survey seemed to be of no use after discarding Spacy. The dataset merely consisted of resumes annotated by an online tool which added a lot of ambiguity to the initial part of our pipeline which involves extracting the text from a PDF file. Conversion of this extracted text to the format used in the dataset was beyond the scope of the goals of our project. An issue during the literature survey report was the inability to measure the model's performance. The evaluation had to be done manually which was untypical of for any learning based process.

In addition to all of this, there was confusion regarding the method of classification for our model. Over the course of time, with a lot of reading, analyzing and trying few methods out, we decided to stick to supervised methods for learning. The selection of the model under supervised learning would be discussed later in the report.

### B. Conclusions from the Literature Review

- After the initial model was implemented and tested , for all of the above mentioned reason the process had to be done from scratch.

- This time around, the approach was to build the pipeline piece by piece from scratch and perfect it to the needs of chosen problem statement.
- The problem statement was refined to a certain extent such that the course of procedures become clear.
- The evaluation parameters became clear with the problem being a clear classification problem.

The new steps taken and the detailed procedure is explained later in the report.

## III. PROBLEM STATEMENT

Every organization has to deal with folders together of resumes. Going through these resumes can be a tiring process added to the fact that is its very time consuming. It would help a ton if there were to exist a model which processes the resumes and not only gives out how many and which of them meet the requirements, but also gives a summarized version of the resumes. These concise resumes can come of great use for a hiring company in their selection procedures , to straight away reject those applications that are not suitable for the job description. It can be extended to a SPAM finder for Resumes.

So the refined problem statement stated would be " Classification of Resumes into job domains with a summarized report of the resumes using NLP techniques"

### A. Data Source

Our dataset is a set of 1219 resumes, each labelled with a category that it most suitedly belongs to. These resumes can be divided into 25 classes. The dataset is pretty simple yet noisy due to a lot of UTF-8 characters and censored content for privacy reasons. There were also records which were wrongly formatted and hence had to be dropped. The dataset has 3 columns:

1) ID which contains id for each row.
2) Category which is the domain/work area in which resume lies.
3) Resume which is the text of resume.

### B. Assumptions and Constraints

- Our model expects the resumes in PDF format , which is the hardest conversion process that a resume could go through until it is merely a string. We chose to stick to this because in alomst all instanaces, resumes are found in pdf format itslef. Morover, Word documents can be easily converted to text without requiring any major concepts like OCR.
- The input resume is assumed to be written in English since OCR and pre-processing like removal of stop words is strictly limited to English Script.

## IV. PROPOSED PIPELINE

Our pipeline can be divided into 5 stages. A diagrammatic representation of out project pipeline is given in Fig 1:


Fig. 1. proposed pipeline

As shown in the block diagram, our entire work flow can be divided into Data Pre processing, Vectorization of text, Training and comparing different models for classification, Selection of the best model, and finally, Summarization.

### A. Pre Processing

1) PDF to Text:
   - Our aim for this project is to come up with an end-to-end tool which takes in a document and gives out the expected result, in this case - The category and the summary. The majority of the resumes out there are submitted in pdf format, we decided to add a preprocessing step of converting PDF to text, by making use of the well known Optical Character Recognition. We made use of pdfminer under python for this task.
2) Data Cleaning :
   - *Conversion of encoding:* The Resumes are in byte strings encoded in utf-8 format. They seem to include many utf specific character codes, which aren't interpretable in ascii and can add a lot of noise in assessing the resumes too. To tackle this, thorough cleaning was carried out along with conversion to ASCII.
   - *Censored Information*: Since the dataset had a collection of real-world resumes, lot of personal details like phone numbers, email ids, address etc had been censored (replaced by a string of 'x') for privacy reasons. eg: email:xxxx.xxxx@xxxx.xxxx . This would add unnecessary noise to the dataset as it adds no value whatsoever.
   - *Unnecessary separators:* A lot of resumes had separators like a string of '-', which was considered to be removed too.
   - *Punctuation and Stop Words*: Punctuation and stop words didn't seem to add any value to the analysis, and hence it was decided to be gotten rid of.
   - *Erroneous Formatting*: There were also some records with highly erroneous formatting which came in the way of our cleaning/analysis. Getting rid of them was the best resort.
   - *Personal details*: Details like email id, phone numbers, dates etc would add nothing but plain noise to the analysis which would add merely any value in the process of classification.It was hence considered best to remove them.
3) Class Labels : For added ease during further processing, mapping the class labels to numeric constants was done. This made it easier for the model to predict.

### B. Vector Processing of Text

- The classifiers and learning algorithms can not directly process the text documents in their original form, as

most of them expect numerical feature vectors with a fixed size rather than raw text docs with variable length. Therefore, during this step, the text is converted to a more manageable representation.
- One common approach for extracting features from text is to use the bag of words model: a model where for each document, a resume in our case, the presence (and often the frequency) of words is taken into consideration, but the order in which they occur is ignored.
- Term Frequency and Inverse Document Frequency is used for each document to create the vectors that represent the resume.

To test if this step was useful , a Chi squared test was done to check the correlation between the vectored terms within a single category. This provided us with some good insights and also allowed us to test on whether words had to be looked at individually or in groups.

Looking at words in groups is the N Grams approach. During the previous tests and some trial and error , the value of n was found to be best suited for the data at 2. That is a Bi-Grams model was chosen.

So on the whole the Bag-Of-Bi Grams representation to represent the frequency count was adopted. This was later fed to a term frequency - inverse document frequency transformer. The tf-idf vectors were then used as training data for all of the further classification models.

### C. Input Representation for models

Given below are the various input representations made use of:

- Word Counts with Count Vectorizer: It provides a simple way to tokenize a collection of documents, built a vocabulary of known words, and to encode new documents using that vocabulary.
- Word Frequencies with Tf-Idf Vectorizer: Term frequency summarizes how often a given word occurs within a document and Inverse document frequency down scales words that appear a lot across documents. The Tf-Idf Vectorizer will tokenize documents, learn the vocabulary and inverse document frequency weightings, and allow you to encode new documents.

We first used a Count Vectorizer and used Tf-Idf transformer to just calculate the Idf and start encoding documents. We trained our classifier on the Tf-Idf vectors. The classifier was then tested on the test documents and also on entirely new and unseen resumes. The selection of a suitable classifier will be explained elaborately in the next section.

### D. Summarization

Summarization is an important part of our tool. The summarization technique that we have used here marks use of a **lexical token summarizer**. It is a summarizer purely based out of scoring the sentences based on the words on the sentences.

*Algorithm*

- Firstly, Sentence tokenization on the resume is carried out.
- A frequency distribution of the words across the the resume at hand is generated. It is to be noted that prior to this step, removal of punctuation marks was done to prevent unnecessary noise.
- This frequency table was used to score the sentence.
- The score of each sentence = Sum of the scores of each word in the sentence.
- The score of a word is the frequency count of the word in the resume.
- Finally, the summarized text is obtained by extracting the most significant sentences in the resume, based on the scores.

This method of summarization worked fairly well, and succeeded in being able to get rid of all the clutter and filler text from the resume and focus only on the significant parts of it.

## V. Experiments

### A. Model Selection

The Naive Bayes model is simple and effective and should be the first method to try on a classification problem. It is a supervised learning approach that uses probabilities of each attribute belonging to each class to make a prediction. Hence we chose this model to make category predictions and the accuracy came up to around 40%.

```
Accuracy from Naive bayes: 0.3875
```

```
print(sklearn.metrics.classification_report(a.Category, a.predicted))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Accountant | 0.64 | 0.64 | 0.64 | 11 |
| Advocate | 1.00 | 0.58 | 0.74 | 12 |
| Agricultural | 0.00 | 0.00 | 0.00 | 8 |
| Apparel | 0.00 | 0.00 | 0.00 | 5 |
| Architects | 0.00 | 0.00 | 0.00 | 2 |
| Arts | 0.00 | 0.00 | 0.00 | 9 |
| Automobile | 0.00 | 0.00 | 0.00 | 7 |
| Aviation | 0.00 | 0.00 | 0.00 | 3 |
| BPO | 0.00 | 0.00 | 0.00 | 2 |
| Banking | 1.00 | 0.14 | 0.25 | 14 |
| Building & Construction | 0.00 | 0.00 | 0.00 | 5 |
| Business Development | 0.00 | 0.00 | 0.00 | 9 |
| Consultant | 0.00 | 0.00 | 0.00 | 6 |
| Designing | 0.00 | 0.00 | 0.00 | 12 |
| Digital Media | 0.50 | 0.11 | 0.18 | 9 |
| Education | 0.75 | 0.82 | 0.78 | 22 |
| Engineering | 0.17 | 0.92 | 0.29 | 24 |
| Finance | 1.00 | 0.43 | 0.60 | 14 |
| Food & Beverages | 0.00 | 0.00 | 0.00 | 3 |
| HR | 0.80 | 0.80 | 0.80 | 5 |
| Health & Fitness | 0.71 | 0.45 | 0.56 | 11 |
| Information Technology | 0.44 | 0.55 | 0.49 | 20 |
| Managment | 0.33 | 0.31 | 0.32 | 16 |
| Public Relations | 0.00 | 0.00 | 0.00 | 4 |
| Sales | 0.83 | 0.71 | 0.77 | 7 |
| avg / total | 0.43 | 0.39 | 0.35 | 240 |

Fig. 2. Naive Bayes Accuracy

### B. Comparison with other models

Since the accuracy of Naive Bayes wasn't satisfactory, we moved on to compare it with other models to find a better one. The models we used to compare the results were Support Vector Classifier, Logistic Regression and Random Forest. Amongst all the different models Random forest algorithm tends to perform worse than Naive Bayes and SVC performs the best with an average accuracy of 70%-75%. Fig 3 is a multi BoxPlot showing how the models performed.

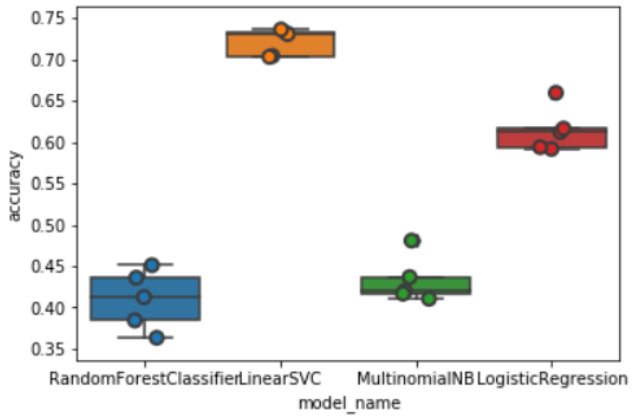


Fig. 3. Comparison of model

The SVC model from our inference must have performed the best merely due to the topology of this dataset. It does a better job of capturing non linearities than Bayes or Logistic regression. Also the SVM model originated from an optimization problem. This makes it work better than most other models on multivariate numeric data , which is the case with our vector representation of words. Below is a diagram of how SVC performed on our dataset.

## VI. A QUICK WALKTHROUGH OF OUR END MODEL

We have explained the steps we took to develop our model. Now, we would like to explain how our project would work as a finished tool.

- The input to the tool would be a PDF document. One doesn't have to worry about the encoding as it will be all be taken care of by our model.
- Our model extracts text from the resume using OCR.
- Clean the text using our user defined cleaning function to get clean and plain text.
- Pass this through our trained SVM classifier to get the result ie., the class of resume.
- Pass the resume to the (Text of resume + Sentence tokenized version of the resume) to our summarizer, to get the final summary.

All this happen inside our granular tools and all the user has to do is give the PDF document and wait for the model to give out the class and the summarized version of the resume for him to refer to based on his requirements!!

```
Accuracy from SVM: 0.713888888889
```

```
print(sklearn.metrics.classification_report(a.Category, a.predicted))
```

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Accountant | 0.82 | 0.78 | 0.80 | 23 |
| Advocate | 1.00 | 0.92 | 0.96 | 26 |
| Agricultural | 1.00 | 1.00 | 1.00 | 5 |
| Apparel | 0.00 | 0.00 | 0.00 | 4 |
| Architects | 1.00 | 0.25 | 0.40 | 4 |
| Arts | 0.87 | 0.87 | 0.87 | 15 |
| Automobile | 1.00 | 0.44 | 0.62 | 9 |
| Aviation | 1.00 | 1.00 | 1.00 | 3 |
| BPO | 1.00 | 0.67 | 0.80 | 9 |
| Banking | 0.83 | 0.42 | 0.56 | 12 |
| Building & Construction | 0.80 | 0.33 | 0.47 | 12 |
| Business Development | 0.58 | 0.47 | 0.52 | 15 |
| Consultant | 0.00 | 0.00 | 0.00 | 7 |
| Designing | 0.62 | 0.62 | 0.62 | 16 |
| Digital Media | 0.91 | 0.62 | 0.74 | 16 |
| Education | 0.65 | 0.85 | 0.73 | 26 |
| Engineering | 0.69 | 0.84 | 0.76 | 37 |
| Finance | 0.71 | 0.60 | 0.65 | 20 |
| Food & Beverages | 1.00 | 0.80 | 0.89 | 5 |
| HR | 0.87 | 0.93 | 0.90 | 14 |
| Health & Fitness | 0.83 | 1.00 | 0.90 | 19 |
| Information Technology | 0.59 | 0.79 | 0.68 | 29 |
| Managment | 0.39 | 0.52 | 0.45 | 21 |
| Sales | 0.55 | 0.92 | 0.69 | 13 |
| avg / total | 0.73 | 0.71 | 0.70 | 360 |

Fig. 4. SVM Accuracy


Fig. 5. Steps of execution of the combined tool

## VII. CONCLUSIONS

A tool like the one we developed can potentially speed up the process of recruitment and also bring down the time and effort. Through the result analysis that we have done, we see that the model performs quite well, and is seen to predict the primary category of the resumes quite accurately. Using this information, the recruiter can easily narrow down the set of resumes that he needs to look into for further activities like technical interviews, group discussions etc. By referring to our summarized resume,which is expected to cut down on the filler text, the recruiter can further ease his task of going through the resume by just focusing on the important aspects.

The future for a tool like this is bright! It has the potential to completely automatize the process of recruitment.

## ACKNOWLEDGMENT

We would like to thank our Data Analytics teacher, Dr. Gowri Srinivasa for her constant support throughout the project. We would also like to thank PES University for this opportunity.

## REFERENCES

[1] Sonar, Swapnil, and Bhagwan Bankar. Resume parsing with named entity clustering algorithm. paper, SVPM College of Engineering Baramati, Maharashtra, India (2012).

[2] Sanyal, Satyaki, et al. Resume Parser with Natural Language Processing. International Journal of Engineering Science 4484 (2017).

[3] Mehdi Allahyari,Seyedamin Pouriyeh, Mehdi Assefi, Saied Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut - A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques KDD Bigdas, Halifax, Canada (August 2017).

[4] Natural Language Processing Made Easy using SpaCy (in Python)-https://www.analyticsvidhya.com/blog/2017/04/naturallanguage-processing-made-easy-using-spacy-in-python/(April 2017)

[5] "Processing Pipeline - https://spacy.io/usage/processingpipelines/custom-components

[6] Document Classification -https://en.wikipedia.org/wiki/Document classification

[7] Text Classification is Your New Secret Weapon - https://medium.com/@ageitgey/text-classification-is-your-new-secretweapon-7ca4fad15788

[8] "Processing Pipelines"- https://spacy.io/usage/processing-pipelines

[9] "Using Machine Learning to Retrieve Relevant CVs Based on Job Description"-https://dzone.com/articles/cv-r-cvs-retrieval-system-based-on-job-description

[10] "Information Extraction from documents"-https://medium.com/@divalicious.priya/information-extraction-from-cv-acec216c3f48

[11] "Training NLTK models on python"-https://blog.sicara.com/train-ner-model-with-nltk-stanford-tagger-english-french-german-6d90573a9486

[12] "Using Deep Learning To Extract Knowledge From Job Descriptions"-https://www.kdnuggets.com/2017/05/deep-learning-extract-knowledge-job-descriptions.html

[13] "Gentle Introduction Bag Words Model"-https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[14] "How to process textual data using TF-IDF in Python"-https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3

[15] "Document Classification"-https://www.kdnuggets.com/2015/01/text-analysis-101-document-classification.html

[16] Joachims, Thorsten. Learning to classify text using support vector machines: Methods, theory and algorithms. Vol. 186. Norwell: Kluwer Academic Publishers, 2002.

[17] Nigam, Kamal, et al. "Text classification from labeled and unlabeled documents using EM." Machine learning 39.2-3 (2000): 103-134.

[18] "How To Implement Naive Bayes From Scratch in Python"-https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

[19] "Support Vector Machine"-https://scikitlearn.org/stable/modules/svm.html

## VIII. CONTRIBUTIONS

1) Manish Shetty M:
   - Pdf to Text using pdfminer
   - Vector representation using tf-idf and bag-of-words approach
   - Chi-Square test for correlation within categories
   - Comparison of 4 models
2) Megha K:
   - Naive Bayes classifier
   - SVM classifier
   - One simple application of the tool.
3) Meghana:
   - Data Cleaning
   - Exploratory Data Analysis on the dataset
   - Class-wise most frequent Bigrams
   - Text Summarization