Resume Parsing and Ranking using NLP & NER

Meghana [1],Megha Kalal[2] and Manish Shetty M[3]

***Abstract*— This paper gives an outlook on a modelled pipeline to parse , summarize and rank resumes. The resume parser automatically segregates information on the basis of various fields and parameters like name, email , job description , work experience , phone / mobile nos. etc. Input to train the spaCy model is an annotated data set of resumes in JSON format . The spaCy model used for parsing and summarizing the resumes learns the important features and thus can generalize to any newly given resume. A classification model groups the resumes into classes that represent the user's requirement , such that all resumes within a class are highly similar with respect to a job standpoint. Finally a ranking algorithm scores the resumes within a class for each parameter which is then used to rank the resume.**

I. INTRODUCTION

Large companies and recruitment agencies receive, process and manage hundreds of resumes from job applicants. Besides, many people publish their resumes on the web. Dealing with loads of resumes at once can be time consuming since all they need is a bunch of valuable resumes that represent candidates specialized in fields the company/agency is looking for. The resumes that one receives in their original format (pdf, docm etc.) are unstructured. The unstructured format of these resumes , with random templates and fonts make it difficult for processing. These resumes can be automatically retrieved and processed by a resume information extraction system. Extracted information such as name, phone / mobile nos., e-mails id., qualification, experience, skill sets etc. can be stored as structured information in a database. There are 4 main methods used for resume information extraction

- Named Entity based which try to identify Certain words , phrases and patterns using either regular expressions or dictionaries.
- Rule based which are controlled by large no of grammatical rules
- Statistical based which apply numerical models
- Learning based which employ classification algorithms to extract information.

The Ranking is based on scores given to segmented components and calculating an overall score based on the given requirements of the user.

II. PREVIOUS WORK

A. Commercial Products

Recent advances in information technology such as Information Extraction (IE) provide dramatic improvements in conversion of the overflow of raw textual information into structured data which constitute the input for discovering more complex patterns in textual data collections. Resume information extraction, also called resume parsing, enables extraction of relevant information from resumes which have relatively structured form. Although, there are many commercial products on resume information extraction, some of the commercial products include Sovren Resume/CV Parser , Akken Staffing , ALEX Resume parsing , ResumeGrabber Suite.

B. Research Publications

1) Resume Parser with Named Entity Clustering[1] : This paper presents a resume information extraction system based on Named Entity Clustering Algorithm . The resume extraction process consists of 4 phases. In the first step, a resume is segmented into blocks according to their information types. In the second step, named entities are found using special chunkers for each information type. In the third step, the found named entities are clustered into groups according to their distance in text and information type. In the fourth step, normalization methods are applied to the text.
2) Resume Parser with Natural Language Processing[2]: This paper discusses segmentation, parts of speech tagging , tokenizing and the systematic steps to be followed to parse a resume. The system was also able to scrape from different social networking sites including Stack Overflow, LinkedIn, etc and find the similarity between them with which we could determine the genre of the resume (e.g: Computer science, Management, Sales, human resource, etc). Future work was listed to include ranking the resume and analyzing information about the candidate from social networking sites like Face book and Twitter so that they can decide more accurately and authentically whether or not to offer the candidate a job.

C. Approach and Results

Both the described and the other cited publications have limited their approach to summarizing the resumes into easily documented formats for database management. Typically, one would go about the below mentioned sequence of steps, simply put, a 'pipeline' to finally arrive at a solution to the problem.

- Preprocessing
    1) Data cleaning , integration , transformation and reduction
    2) Tokenization : Tokens are usually referred to as terms or words, but sometimes fabricating a type/token distinction is essential. A specimen of an array of characters in a document that is assembled as a helpful acceptable unit for processing is

called a token. Whereas, the group of tokens which consists of same character sequence is called type. A type that is added to the dictionary of IR system is called term

3) Grouping of inflected forms of a word - Stemming and Lemmatization : Words that mean the same can be used in different inflections in sentences. Simplifying them down to their most unconjugated version plays a vital role in NLP. A heuristic approach to this would be 'Stemming' where ends of words are chopped off in hope of achieving the basic form. Lemmatization, on the other hand, does the task more systematically using vocabulary and morphological analysis of words.
4) Part-Of-Speech tagging : Assigning a grammatical entity (also called 'part of speech') to every token. It is also known as grammatical tagging or word-category disambiguation.

- Modelling using NLP
  1) Lexical Analyzer : Using a database / dictionary of words most commonly found in resumes .Now when a new resume taken, the parser searches for the keywords and extracts all the data between the starting and the ending of them, which we call as segments. Named Entity Recognizer is used to extract data from each segment specifically. This method makes the system efficient and reduces its complexity.
  2) Syntactic Analyzer : Parse tree generated with hierarchy of words
  3) Semantic Analyzer : This process relates syntactic structure to the level of the writing as a whole from the levels of clauses, phrases, paragraph and sentences.

D. Lacunae

1) The publications and other projects all limit their outcomes to simply summarizing the resumes.
2) The evaluation criteria used is completely based on how good NER performed and not the model of ranking any resume over another.
3) The models implemented have not been able to classify the resumes into their respective field of jobs or any other parameter , which could have been used in the ranking system.

III. PROPOSED WORK

As mentioned above, not much has been explored beyond just summarizing a resume to yield nothing but a shortened version of the same. While this is seen to make things better, our approach is expected to further ease the laborious task of going through the resumes by actually ranking the resumes based on what the user expects from the candidates and what makes a particular candidate stand out; in a a nutshell - highlight the 'ups and downs' of hiring the candidate in question.

Our entire work-flow can be divided into two segments:

A. Extract the significant parts of the Original documents - Summarization:

Out of a wide variety of options available, we narrowed our choices down to 'Stanford NER' and 'spaCy'. Stanford NER is a library implemented in Java, while also available for python. After a detailed analysis of both of them, we decided to work with spaCy. Where Stanford NER fails is when retrieval of entities comes into picture. The task becomes cumbersome when the number of words to consider crosses one. Adding to this, the performance of tagging is the slower when compared to spaCy. spaCy, on the other hand, seems to do a better job at getting the entities with no extra setup required. Furthermore, it also supports GPU and deep learning which could be considered for future works.

1) spaCy for Advanced NLP: spaCy is an open source library for advanced NLP in Python. spaCy provides a concise API to access its methods and properties governed by trained machine (and deep) learning models. Some of spaCy's features and capabilities are Tokenization, POS tagging, Dependency Parsing, Lemmatization, Sentence Boundary Detection, Named Entity Recognition, Similarity, Text Classification etc. spaCy provides a variety of linguistic annotations to give you insights into a text's grammatical structure. Given a particular text, spaCy first tokenizes the text to produce a Doc object. The Doc is then processed in several different steps, this is also referred to as the processing pipeline.

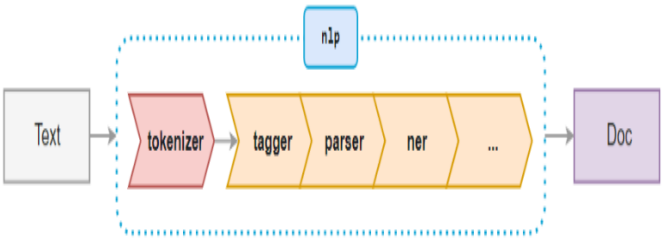

Fig. 1. spaCy pipeline

2) Processing Pipeline[4]: The processing pipeline always depends on the statistical model and its capabilities. The pipeline used by default models consists of a tagger, parser and an entity recognizer.

1) Tokenizer: Every spaCy document is tokenized into sentences and further into tokens which can be accessed by iterating the document.
2) Tagger: Part-of-speech tags are the properties of the word that are defined by the usage of the word in the grammatically correct sentence. These tags can be used as the text features in information filtering, statistical models, and rule based parsing.
3) Dependency parser: One of the most powerful feature of spaCy is the extremely fast and accurate syntactic

dependency parser which is used for sentence boundary detection and phrase chunking.

4) Entity Detection: spaCy consists of a fast entity recognition model which is capable of identifying entity phrases from the document. Entities can be of different types, such as person, location, organization, dates, numerals, etc.

Custom components can also be added to the pipeline. By adding a component to the pipeline, you'll get access to the Doc at any point during processing instead of only being able to modify it afterwards.

B. Document Classification and Ranking:

Following the summarization, we go to the next segment - Building our 'Classification Model' . On a general note, Document Classification can be carried out in two ways[6]:

- Content Based Classification: The approach here is "Weight-based" ie., weights are assigned to the subjects/sections of the resume and it is using these weights that the documents are assigned their class.
- Request Based Classification: This approach is also call "Indexing". Here, the user has a say in what needs to be looked into for classifying the document. This approach would come to use in situations where the fields to analyze changes with the target population.

Our classification model will learn the important features of a resume that can help classify them based on a Job Description required by the user (ie: Programmer , Banking , Activist , Research , HR etc). The problem can be viewed in two ways:

- Binary Classification: Here the classifier would be trained to classify each resume as "Suitable" or "Not Suitable" based on the user's requirements.
- Multilevel Classification: Here, the input resumes would simply be classified into the class our classifier think it best fits into to finally give us a distribution of all the resumes across the classes. The user then needs to specify which class of resumes he/she requires.

Irrespective of the two methods , the steps to follow remain the same. As for the method of classification, we plan to make use of "Content based" approach.

- The first step to classification would be to create a corpus of words that we will most likely be seen in resumes. Since the analysis will be done on weights , each word will require a "word embedding" i.e: a vector that represents the word in the required no of dimensions. Word2Vec format will be used at this step.
- To do the above task , we will need to create a hybrid word embedding space using an existing word embedding space by making use of dimensionality reduction methods like PCA.
- At this stage the model can be trained over these weighted vectors using[3] supervised learning models like Logistic Regression or unsupervised learning models like K-means Clustering , where we finally choose the model that fits best.

For purpose of ranking the resumes within a class a simple algorithmic approach will be followed. Grades given to each resume within certain chosen parameters combined with the probability of the resume being "suitable" to that class will then aggregate onto a resulting overall rank for the resume in it's class.

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/natural-language-processing-made-easy-using-spacy-in-python/(April 2017)

[5] "Language Processing Pipeline" - https://spacy.io/usage/processing-pipelines/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-secret-weapon-7ca4fad15788