International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

"ATS BREAKER"- A System for Comparing Candidate Resume and Company Requirements

Lino Mathew
Dept. of Computer Science and Engineering
Amal Jyothi College of Engineering Kottayam,
Kerala, India

Nikitha Linet
Dept. of Computer Science and Engineering
Amal Jyothi College of Engineering Kottayam,
Kerala, India

Nithin C George
Dept. of Computer Science and Engineering
Amal Jyothi College of Engineering Kottayam,
Kerala, India

Nithin K Thomas
Dept. of Computer Science and Engineering
Amal Jyothi College of Engineering Kottayam,
Kerala, India

***Abstract*— In current scenario, job searching is one of the major challenge faced by people after education. In the job search process, a well-written resume is essential. As the companies receives thousands of resumes, they uses several filtering techniques to identify the most eligible ones according to the company's requirements. This one of the major challenge faced by the candidate in selection process. This paper proposes a model that computes a comparison and based on the results, suggestions are provided to the candidates to modify their resume. The proposed procedure extract information related to technical as well soft skills from the CV's submitted in text, pdf or docx format. The system also provides suggestions for correcting the grammatical errors. The proposed system is designed based on Natural Language Processing (NLP) techniques.**

Keywords*— NLP, NER, text segmentation, React, Flask, LanguageTool

I. INTRODUCTION

After completing education, mostly jobs are the next phase that comes in a person's life. However, there are lots of people who start working before completing their formal education. In this era of technology, job searching has become more smart and easier at the same time. While searching for jobs the most important thing to represent an applicant is Curriculum Vitae (CV) or Resume. In the candidate screening process, the first level is filtering of candidates using their resume and then followed by an interview. However, there are more than enough applicants for a single job and it is really tough for an employer to select candidates only based on their CV / Resume. To solve this problem, most of the company's provide their requirements and there has always been an attempt to use an automated method where the employers can easily select qualified candidates in a short time. For this most of the companies use an ATS (Applicant Tracking System), which are software that the organizations use to figure out the important parts or the points of interest in the CV/Resumes and ignores the rest. The innovation in the field of Natural Language Processing along with Machine Learning [1] has been really helpful in this case. The ability to understand unstructured written language and extract important information from it to teach the machine is exactly what is needed to analyze any written documents such as resume papers just like human being. Natural language processing is the field of machine learning that allows the computers to recognize the meaning of human languages. There are different ways to deal with training a model [2] and tackle issues. Some of the major tasks in natural language processing are automatic summarization, translation, named entity recognition [3][4], parsing, information extraction, information retrieval etc.

This journal proposes a system that automatically compares a candidate's resume with the company requirements. The idea of this system is that it helps the candidate to have prior knowledge about the chances of him/her qualifying the filtering level. Also the candidate can use the suggestions provided for modifying their resume. The second part of the paper contains a detailed description of the proposed system. The third part describes the training deals for extracting the skills from the resume and how they are categorized. In addition, the fourth contains the details about the tools used.

II. FRAMEWORK

Our objective is to build a model that would provide suggestions to the candidates based on the comparison results obtained. The proposed system extracts information about work experience, education, basic details etc. from the candidates resume and job requirements from the company's demand. The obtained data is used for comparison, and results are obtained. Based on this result, suggestions are provided to the user. These suggestions are based on the percentage obtained. It also finds the grammatical mistakes in the resume. Suggestions to correct the mistakes are also provided to the user.

For availing the above features, the user has to create an account using their name, email address and a password. The user can use this account for further uses. All the comparisons and resumes created are stored in the database so that user can utilize these results for future purposes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

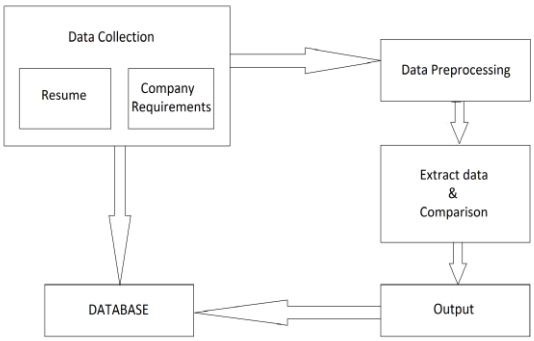

Figure 1: Proposed System Framework.

Figure 1, shows the framework for the proposed system. It primarily consists of 4 steps:

STEP 1: Data Collection
STEP 2: Data Preprocessing
STEP 3: Data Extraction and Comparison
STEP 4: Display the output and storage

A. Data Collection.

The user creates an account using name, email address and a password. These details are stored in the database which is later used for user authentication. Passwords are stored in the form of hash values for security purposes. Once the user logs in, they have to provide their resume and company's job description as input. The input file can be in the form of text, PDF, docx, etc. The user end is developed using React.js [5] which creates a very interactive and dynamic user interface.

B. Data Preprocessing

The input from the user can be in the form of text, pdf or docx. This is converted into text format. PDFMiner, a tool to extract data [6] from PDF documents, is used. The data from the docx file is extracted using doc'x'2txt. The output text format is fed into the model for extraction and comparison.

C. Data extraction and comparison

Once the preprocessing is done, the text file is provided as input to the model which is trained using spaCy [7]. The details on training the model is explained in the next section. The model extracts the skills from the candidate's resume and company's requirements. While extracting, if the model identifies an untrained skill, it is categorized using regular expressions [8]. When a new phrase is encountered, the phrases before it is analyzed. If the previous phrase is a proposition and newly encountered phrase is a noun, then the noun phrase is identified as a skill. The copy of the newly identified skill is stored to the trained model and output to the user. The extracted data is stored into two different dictionaries. This extracted data is used for making comparisons and provides the user with a match percentage.

D. Displaying the output and storage

The computed results are displayed to the user in the form of a table which consists of three columns. The first column represents the skills extracted, second and third column represents the word count from resume and company requirements respectively. An overall match percentage is also displayed. Based on this, suggestions are provided to the user to modify the resume. Suggestions can be for grammar corrections, skills modifications etc. LanguageTool API [9] is used to identify grammar mistakes and based on this, suggestions for correcting them are prompted to the user. Skill modification suggestions include adding missing skills, specializations etc. All the results are stored in the database, so that the user can refer or use them for later purposes. The candidate resume and company requirement is stored using a file system.

III. TRAINING THE MODEL

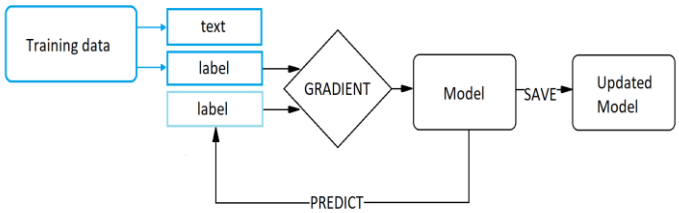

Figure 2: Training the model.

Figure 2.,shows how a spacy module trains the NER model. Spacy provides a default model which has the ability to understand a wide domain of named or numerical entities. Since in our project we need to recognize skills, which may not be a predefined entity in spacy, we will need to add this entity to the NER model. This can be done by training the model, so that it updates the model with newer trained examples. The dataset for the training purpose consists of 150 resumes and the file will be provided in .json format. We need to convert the data which is in .json format to a format which is supported by spacy. This converted data will be used for the training purpose. Spacy's models are statistical and every decision they make is a prediction. Decisions regarding which part-of-speech tag is to be assigned or whether a word is a named entity or not, such decisions are all considered as predictions. These are based on the examples that the model has seen during training. It works in such a way that, when the model is provided an unlabeled text it will make a prediction.

The feedback about the accuracy of the prediction is provided to the model is in the form of an error gradient. The feedback can be given because we know the correct answer. This error gradient is used to calculate the loss function. Loss function is the difference between the training example and the expected output. As the difference increases, the greater will be the significance of gradient

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

and updates to our model. This will help our model to define a theory that can be used to generalize other examples. We don't want our model to just remember the examples, instead we will train our model to a number of iterations. Also we make sure that at each iteration, the order of training data is changed to make sure that our model doesn't make any generalizations based on the order of the data. We use nlp.update method to update our model after each iteration. This method steps through the words of the input. Finally we can save the trained data to the directory using nlp.to_disk method.

When the user inputs the resume and job description in the prescribed columns, we need to extract the skills from both of these. The user may input the resume as in PDF format so we need to convert this to text format. From this we extract the skills using some NER methods present in the spacy. We will load our previously trained model for extraction. As we enter a new resume it will go through the resume and find all the skills present in it. The same steps will be done for the job description also. Finally we will obtain all the skills present in both resume and job description entered by the user. There may be a chance for our model to encounter some untrained skills during extraction. Naturally they will not be extracted, but we try to extract those untrained skills too using some rules and regular expressions. Rule Based Matching in spacy [10] helps to achieve this. This feature allows us to match tokens, phrases and entities of words and sentences using some preset patterns. It also supports regular expression matching. We can set up some rules or patterns to recognize the untrained skills in our resume or job description. For that we require the skills which have been extracted using our trained model. These extracted skills are used to check whether it follows a separator (comma, or, and) and then we will check the parts of speech of the next word. If it is a noun then our model will consider it as a skill and extracts it.

Reference [11] gives two example for the above cases, "In addition, technologies such as SOLR, Mongo DB, Cassandra and Hadoop will be incorporated as well". In this example, suppose SOLR is an extracted skill and the remaining are unknown for our model, then by applying the rule we have mentioned, we can also extract Mongo DB, Cassandra and Hadoop as skills. Another instance where we can apply some rule is when a skill is followed by a preposition para phrase which is followed by a noun phrase, then the noun phrase can be identified as a skill. In addition, open source technologies such as SOLR, Mongo DB, Cassandra and Hadoop will be incorporated as well. In the above example such as is a preposition phrase and it is followed by some Noun phrases. Open source is a skill that has been extracted already by our trained model and it is followed by preposition phrases and then Noun phrases. So the Noun phrases are extracted as skills. We can use these types of various rules and patterns to extract untrained skills. These newly extracted skills will be added to our training model, so that we need not repeat the above procedures when we encounter these skills again. This also helps to update our training model with newer skills.

This trained model can be used to extract the details from the resume. The extracted skills are stored in two different dictionaries. These extracted details are compared and corresponding word counts are calculated. Using this calculation, a match percentage is calculated. Based on the percentage, suggestions to modify the resume are provided to the user.

IV. TOOLS USED

The aim of the project was to apply natural language processing methods and named entity recognition in the ATS Breaker. The comparison model was created using the Python language. The code written in python language is deployed on an application called Heroku. Python is a widely used high-level, dynamic programming language on its own, but with the help of a few popular libraries such as spaCy. Spacy is an open source library for NLP. Several features of spaCy like tokenization, text classification etc., can be used to build systems that extract information.

The system uses Flask, a web framework which connects the user-end and back-end. It uses the SQLAlchemy library to access the read/write operation in PostgreSQL database. The user interface is built using React. The system is deployed in Heroku platform. LanguageTool, a free open source grammar check API is used for grammar checks.

V. PROS AND CONS OF DIFFERENT TOOLS

TOOLS USED	ADVANTAGES	DISADVANTAGES
React JS	-Creates dynamic web applications, -Reusable components, -Easy to learn and use.	-Poor Documentation, -It covers only the user interface.
Flask	-Scalability -Flexibility -High Modularity	-Not Standardized -Fewer number of tools.
PostgreSQL	-Open source -Supports ACID (Atomicity, Consistency, Isolation, Durability) -Contains user defined data types.	-Less popular -Installation is difficult. -Less performance. -Lack of skilled professionals.
Heroku	-Easy to start -Saves time -Allows scaling	-Does not provide IP address.

VI. CONCLUSION

A different way of evaluating and analyzing the data in a CV/Resume is proposed in this system. This is because the system helps the candidates to have prior knowledge about to what extent they have the chances of clearing the screening process based on their resume. The proposed system extracts technical information from the CV and categorizes them for comparison. It also successfully stores

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 06, June-2020

the analyzed results, which the user can refer to for future purpose.

In the future we plan to expand the system by adding a module to generate the resume manually. The same system can be used by the user to generate a resume using the templates provided. This helps the user to keep track of the modifications made based on the suggestions.

ACKNOWLEDGMENT

Firstly, we would like to express gratitude towards God Almighty for his kindness and blessing provided throughout the project. Next we would like to offer our modest thanks to Dept. Software engineering, Amal Jyothi College of Engineering, Kanjirappally for providing us the opportunity to take up this venture. We also like to express gratitude toward Head of Department Mr. Manoj T Joy for his important proposals and collaboration. Next, we would like to offer our genuine thanks to our teachers for their guidance. This project would not have been a great success without the assistance given by our loved ones in all the research work accomplished for the project and the enormous encouragement given to us.

REFERENCES

[1] "Machine Learning: What It Is and Why It Matters," Sas.com. N.p., 2016. Web. 17 Apr. 2016.
[2] McCallum, A., & Nigam, K. (1998, July). "A comparison of event models for naive bayes text classification". In AAAI-98 workshop on learning for text categorization (Vol. 752, pp. 41-48).
[3] Zhou, GuoDong; Su, Jian. 2002. "Named Entity Recognition using an HMM-based Chunk Tagger." Proceedings of the Association for Computational Linguistics (ACL), Philadelphia, July 2002. Laboratories for Information Technology, Singapore.
[4] Andrew Borthwick, "A Maximum Entropy Approach to Named Entity Recognition". Ph.D. Thesis. New York University. September, 1999.
[5] https://reactjs.org/
[6] Ramakrishnan, C., Patnia, A., Hovy, E., & Burns, G. A. (2012). "Layout-aware text extraction from full-text PDF of scientific articles". Source code for biology and medicine, 7(1), 7.
[7] https://spacy.io/api
[8] http://nlp.stanford.edu/software/CRF-NER.shtml.
[9] https://languagetool.org/dev
[10] https://medium.com/rule-based-matching-with-spacy-295b76ca2b68
[11] Thimma Reddy Kalva, "Skill Finder: Automated Job-Resume Matching System ", 2013.
[12] Yu, K., Guan, G., Zhou, M.: "Resume information extraction with cascaded hybrid model". In: ACL 2005: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Morristown, NJ, USA, pp. 499–506, (2005).