REPORT

Task 1: Resume Text Extraction and Comparison

File name =final_resume_scorer.ipynb

a) Data extractionfrom Resume

Used fitz to extract data from resume (inlcuding special characters, symbols,etc.) into a string.

b)Data Preprocessing

Used 4 methods:

i)Lowercasing

Converted all the text into lowercase

ii)Removing special characters

Removed unneccessary special characters

iii)Removing stopwords

Removed stopwords that may render In less accuracy of the model

iv)Lemmatization

Lemmatized all the processed data to their root form for generalisation.

c)Modelling

The provided code defines two functions, compare_resume_to_job_description and calculate_suitability_score, which aim to evaluate the suitability of a resume for a given job description. The primary metric used for assessment is the similarity between the content of the resume and the keywords present in the job description.

Algorithm:

compare_resume_to_job_description Function:

Input: resume_content (text of the resume), job_description (list of keywords from the job description).

Output: similarity_score (a measure of similarity between the resume and the job description).

Algorithm Steps:

Initialize num_matching_keywords to 0.

Iterate through each keyword in the job_description.

Check if the keyword appears in the resume_content.

If it does, increment num_matching_keywords.

Calculate similarity_score as the ratio of num_matching_keywords to the total number of keywords in the job_description.

Return similarity_score.

calculate_suitability_score Function:

Input: resume_content (text of the resume), job_description (list of keywords from the job description).

Output: suitability_score (a score indicating the suitability of the resume for the job).

Algorithm Steps:

Call the compare_resume_to_job_description function to get the similarity_score.

Calculate the suitability_score as the similarity_score multiplied by 100.

Return suitability_score.

Usage Example:

Call compare_resume_to_job_description to get the similarity_score.

Call calculate_suitability_score to obtain the overall suitability_score.

Print the calculated suitability_score.

Note:

The code uses a simple keyword matching approach for similarity assessment, and the suitability score is a percentage indicating how well the resume matches the job description.

Further refinement or alternative methods might be needed for a more sophisticated analysis, depending on the specific requirements and characteristics of the resumes and job descriptions.

Task 2: Resume Information Extraction

File name =final_extracting_info.ipynb

a) Data extraction from Resume

Used fitz to extract data from resume (inlcuding special characters, symbols,etc.) into a string.

b)Data Preprocessing

Used 4 methods:

i)Lowercasing

Converted all the text into lowercase

ii)Removing special characters

Removed unneccessary special characters

iii)removing stopwords

Removed stop words that may render In less accuracy of the model

iv)Lemmatization

Lemmatized all the processed data to their root form for generalisation.

c)Algorithms used

i)Extracting Links

The provided code defines a function extract_links_and_emails that is designed to extract URLs (links) and email addresses from a given resume text. The function utilizes regular expressions to identify patterns corresponding to URLs and email addresses within the text.
Summary: The provided code defines a function extract_links_and_emails that is designed to extract URLs (links) and email addresses from a given resume text. The function utilizes regular expressions to identify patterns corresponding to URLs and email addresses within the text.

Algorithm:

extract_links_and_emails Function:

Input: resume_text (the original text of the resume).

Output: A dictionary containing two lists, "Links" and "Emails," representing the extracted URLs and email addresses, respectively.

Algorithm Steps:

Define regex patterns for matching URLs (link_pattern) and email addresses (email_pattern).

Use re.findall to find all matches of the URL pattern in the resume_text. Store the results in a list called links.

Use re.findall to find all matches of the email pattern in the resume_text. Store the results in a list called emails.

Return a dictionary with keys "Links" and "Emails," each associated with its corresponding list (links and emails).

Preprocessing Steps:

The code mentions two preprocessing steps (to_lowercase and lemmatization) before calling extract_links_and_emails:

to_lowercase: Convert the entire resume_text to lowercase. This step ensures uniformity for pattern matching, treating URLs and emails as case-insensitive.

lemmatization: Perform lemmatization on the resume_text. This step might be aimed at reducing words to their base form to improve matching accuracy.

Important Notes:

The code explicitly mentions not taking into consideration the removal of special characters (removing_special_characters) since it may remove important links.

The code also mentions not considering the removal of stopwords (removing_stopwords) because it may decrease the importance of important keywords. This suggests that preserving stopwords is intentional to maintain the relevance of certain terms.

ii)Extracting Skills

The provided code defines a function extract_skills that utilizes the spaCy natural language processing (NLP) library to extract skills from a given resume text. The code performs several preprocessing steps, such as converting the text to lowercase, removing stopwords, removing special characters, and lemmatization. The skills are then extracted by matching the tokens against a predefined list of skills stored in a CSV file.

Algorithm:

Initialize spaCy NLP Model:

Load the spaCy English language model (en_core_web_sm).

extract_skills Function:

Input: resume_text (the original text of the resume).

Output: A list of skills extracted from the resume.

Algorithm Steps:

Apply spaCy NLP processing to the resume_text, creating an nlp_text object.

Tokenize the text, removing stopwords, and store the tokens in the tokens list.

Read the skills from a CSV file (skills.csv). The skills are assumed to be in the column names of the CSV file.

Initialize an empty list called skillset.

Check for one-grams (individual words) in the tokens list that match skills. If found, append them to the skillset.

Check for bi-grams and tri-grams (noun chunks) in the nlp_text that match skills. If found, append them to the skillset.

Return the unique list of skills in title case (capitalized).

Preprocessing Steps:

The code applies several preprocessing steps to the resume_text before calling extract_skills:

to_lowercase: Convert the entire resume_text to lowercase.

removing_stopwords: Remove stopwords from the text.

removing_special_characters: Remove special characters from the text.

lemmatization: Lemmatize the text to reduce words to their base form.

Notes:

The code assumes the existence of functions like to_lowercase, removing_stopwords, removing_special_characters, and lemmatization, but their specific implementations are not provided.

The skills are extracted by matching against a predefined list stored in a CSV file, and the code assumes the format of the CSV file.

iii)Extracting Certifications

The provided code defines a function extract_certifications that uses regular expressions to identify and extract certification information from a given resume text. The certification patterns are assumed to follow a specific format, such as "Certification Name - Year." The extracted information is then returned as a list of dictionaries, where each dictionary represents a certification with "name" and "year" fields.

Algorithm:

extract_certifications Function:

Input: resume_text (the original text of the resume).

Output: A list of dictionaries representing extracted certification information.

Algorithm Steps:

Define a regular expression pattern (cert_pattern) to match common certification patterns in the format "Certification Name - Year."

Use re.findall to find all matches of the certification pattern in the resume_text.

Iterate through the matches and create a list of dictionaries (certifications). For each match, create a dictionary with "name" and "year" fields.

Return the list of certifications.

Preprocessing Steps:

The code applies several preprocessing steps to the resume_text before calling extract_certifications:

to_lowercase: Convert the entire resume_text to lowercase.

removing_stopwords: Remove stopwords from the text.

removing_special_characters: Remove special characters from the text.

lemmatization: Lemmatize the text to reduce words to their base form.

Notes:

The code assumes a specific format for certifications in the resume text, and the regular expression might need adjustment based on the actual patterns observed in the data.

The code does not perform advanced natural language processing (NLP) and relies on pattern matching, which may have limitations based on the variability of certification representations in resumes.