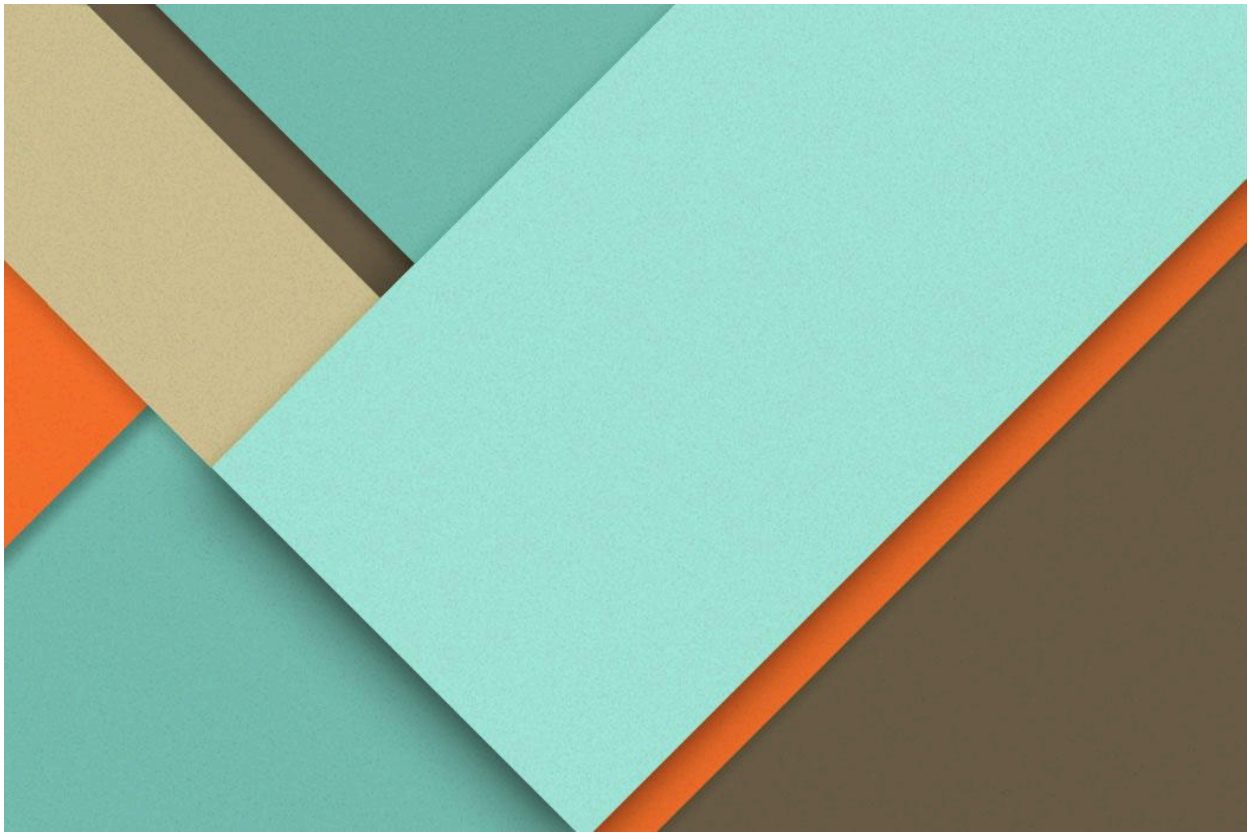


ResumeRevealer

MINED 2024

Dhruv Thakkar

Hetul Shah

Kavya Patel

Prachita Patel

Pranshav Patel

Overview

ResumeRevealer is a comprehensive resume parsing tool developed by Revelio Labs to extract detailed information from resumes in various formats such as PDF, JPG, HTML, DOC, etc. This tool aims to accurately classify text into distinct sections (e.g., education, work experience, skills) and sequence them based on dates, where available. Additionally, it standardizes different job titles and occupations against the O-NET database and extracts detailed skills and competencies from project descriptions and position roles within the resume.

Installation

ResumeRevealer requires several dependencies to be installed. Please follow the steps below to install the necessary dependencies:

1. Install Python: Make sure you have Python installed on your system. You can download Python from the official website.
2. Install Required Packages: Open a terminal or command prompt and run the following commands to install the required Python packages:

   pip install spacy==3.0

   pip install PyMuPDF==1.18.19

   pip install python-docx==0.8.11

   pip install beautifulsoup4==4.9.3

   pip install lxml==4.6.3

   pip install pytesseract==0.3.8

   pip install tensorflow==2.6.0

   pip install sentence_transformers==2.1.0

   pip install transformers==4.11.3

   pip install pandas==1.3.3

   pip install numpy==1.21.2

   pip install Pillow==8.3.2

   pip install scikit-learn==0.24.2

Usage

Follow these steps to use the ResumeRevealer tool:

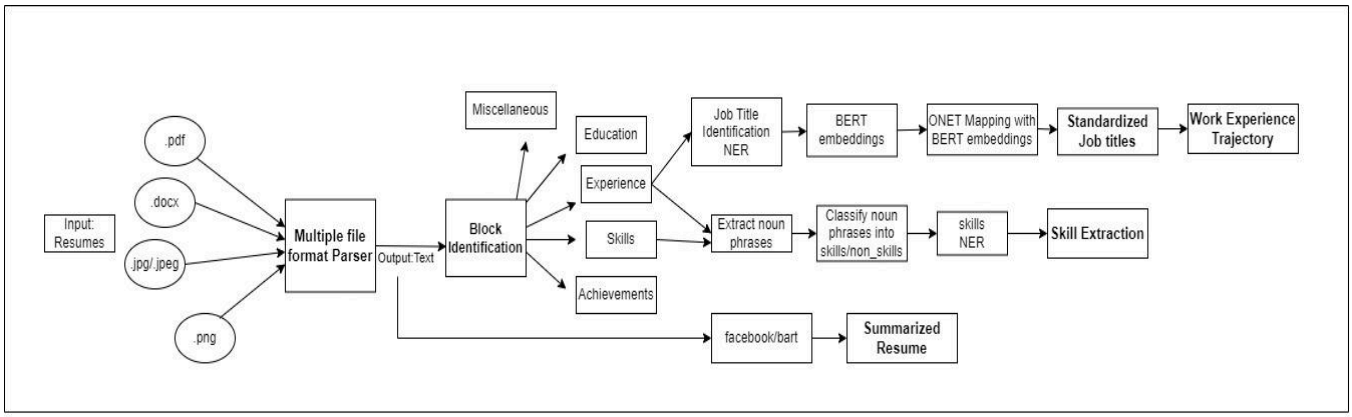
1. Download the Code: Download the code repository containing the ResumeRevealer tool.

   Github Repository:https://github.com/pranshavpatel/MINeD-hackathon

2. Prepare Resumes: Gather resumes in various formats (PDF, JPG, HTML, DOC, etc.) that you want to parse.
3. Run the Parser: Execute the main script to parse the resumes. You can do this by running the following command in your terminal or command prompt:

   python resume_parser.py

PROJECT ARCHITECTURE



1. Resume Parsing

Functionality:

- The project parses resumes in various formats (PDF, JPG, HTML, DOC, etc.) to extract detailed information.
- It accurately classifies text into distinct sections such as education, work experience, skills, etc.

Implementation:

- Utilizes libraries such as PyMuPDF, python-docx, BeautifulSoup, and pytesseract to extract text from different file formats.
- Implements custom logic to classify sections based on keywords and patterns within the text.

PARSING METHOD:

1 entry Filter

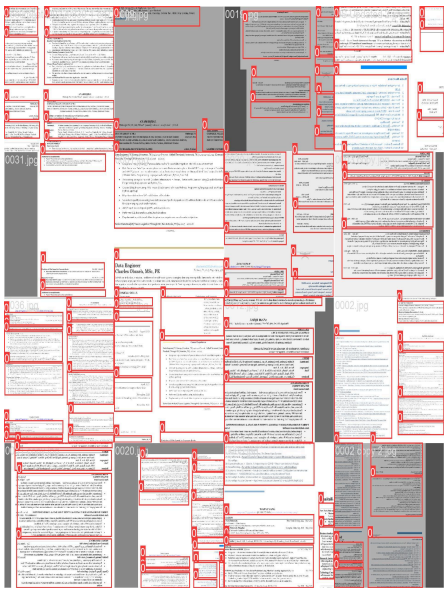
index	skills	experience	education	miscellaneous	achievements
0	software, languages, and tools with the ability to learn new tools, databases and systems to maintain/enhance strategic vision of ,the organization and able to provide cutting-edge data engineering skills. Seeking a reputable organization to contribute to with ,opportunities for personal growth. ,. TECHNICAL SKILLS ,Languages ,• ,Python, SQL ,Others ,• ,Git, Bash , ,Data Manipulation & ,Visualization ,• ,Pandas, Tableau BI, ETL, ,Apache Spark, Hadoop ,HDFS, Snowflake Data ,Factory, SnowSQL and ,Snowpipe, Amazon Kinesis ,Firehose, Databricks, AWS ,EMR, Athena, Lambda, ,Amazon MW Apache ,Airflow and AWS EC2 ,QuickSight and SNS, ,Cloudera infrastructure – ,Ambari, Hortonworks ,Sandbox and Zeppelin ,Notebook , ,Databases and Storage ,• ,AWS RDS, MySQL, ,MongoDB, AWS ,DynamoDB, ,SQLAlchemy, ,PostgreSQL, AWS S3 ,Buckets, SQLite, Azure, ,Oracle. , , , , , , ,Excellent project and time manager with outstanding excellent presentation skills, a great team player and ability to extend ,empathy. , , ,	, , , ,Data Engineer | TC Energy | Houston, TX | 09/2019– Present | Odfjell Terminal | Seabrook, TX | 02/2019– 09/2019 | Enterprise ,Products (Through CF) | Houston, TX | 10/2018 – 02/2019 ,• ,Design and implementation of process data pipeline for asset data migration into AWS cloud environment ,• ,Built Snowflake Data Pipeline using Amazon Kinesis Firehose starting from the AWS EC2 logs to storage in Snowflake ,and AWS S3 bucket post-transformation and orchestrating through Amazon Managed Workflows for Apache Airflow ,(MWAA) DAGs.. Programming language used are Pyspark, Python, SparkSql ,• ,Architecting data pipeline with Cloudera infrastructure – Ambari, Hortonworks Sandbox using Zeppelin Notebook. ,Programming language used are Python, SQL ,• ,Created DataStream using AWS kinesis DataStream and kinesis firehose. Programming language used are Pyspark, ,Python, SparkSql ,• ,Migration solution from SAP and Maximo to Data Lake. ,• ,Solved data quality issues using AWS Databrew and Apache PySpark on AWS EMR and Databricks on AWS EC2 Clusters ,for data wrangling and transformations. ,• ,AWS Project monitoring using AWS Lambda and Aurora ,• ,Performed SQL data analysis using Oracle Database ,• ,Regular meeting with stakeholders to gather user requirements and ascertain objectives. ,Senior Data Analyst | Tesoro Logistics (Through IG) | San Antonio, TX | 04/2017 – 10/2018 , ,• ,Design and implementation of process data pipeline and loading of structured data for performing quantitative and ,qualitative asset risk modelling, predictive analysis for program budgeting and forecast. ,• ,Created process safety risk reduction metrics ,Project Engineer | Weatherford | Houston, TX | 09/2013– 03/2016 ,• ,Performed ETL of drilling well bottom hole pressure '.mbd' data from field and wireline instrumentations (bottom hole ,pressure, well depth and true vertical height of drilling bit). ,• ,Cleaned and transformed data into a '.csv' data file. Load data into Weatherford microflux database for analysis. ,• ,Project management of secure drilling operations of over 80 wells in Canada, Iraq, Nigeria, Angola, and Cameroon for ,clients such as Repsol, Shell, Exxon Mobile, Chevron and Total ,	, , ,University of North Texas | Master of Science | Mechanical and Energy Engineering| Denton, TX ,Rice University | Certificate | Data Analytics | Houston, TX ,	NaN	, , ,Licensed Texas Professional Engineer | Texas Board of Professional Engineer and Land Surveyors ,

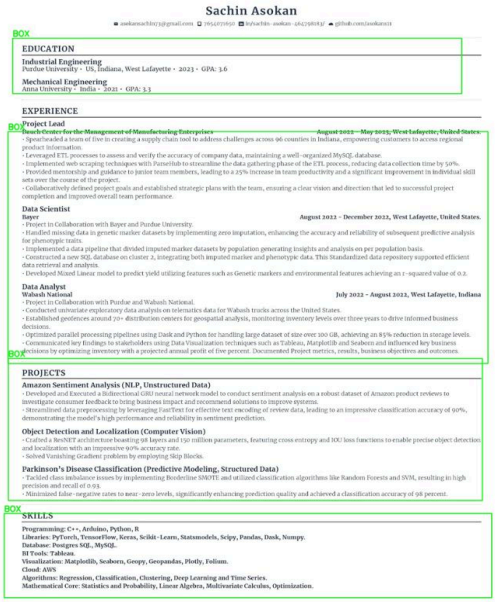
Show 25 per page

- The logic iterates through each line of the text and attempts to match keywords from the predefined section lists.
- If a line contains any of the keywords associated with a section and that section has not been segmented yet (as indicated by the check_segmented list), the code identifies that section as the current section being processed.
- The logic is not computationally expensive compared to language models, so it will be relatively fast
- Additionally, it provides a high level of accuracy, making it an effective tool for achieving reliable results.
- The code also takes care of abstracting subcategories into broad categories

ALTERNATE PARSING METHOD:

The alternate approach involves training a YOLOv8 model on annotated resume images to detect and extract sections such as skills, experience, education, etc. This method utilizes object detection to automatically locate and segment relevant sections within resumes. By training the model on a labeled dataset, it can accurately identify sections across various resume layouts. While this approach offers automation and potentially high accuracy, it requires manual annotation effort to create the training dataset and involves the complexity of training and fine-tuning a deep learning model like YOLOv8.





2. Skill Extraction

1. Extracting Noun Phrases

- The code starts by loading the SpaCy English language model and importing the Pandas library.

- A function called extract_noun_phrases is defined to process text using SpaCy and return a list of noun phrases.
- The final output consists of all the extracted noun phrases stored in the noun_phrases list.

2. Classify Noun Phrases

Classifying Noun Phrases:To address the absence of a relevant dataset, we collected noun phrases from our corpus and manually classified them into two categories: skills and non-skills. This manual classification formed the basis of our training data.

For the classification task, we employed a neural network with the following architecture:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          183700

 conv1d (Conv1D)             (None, 96, 128)           64128

 global_max_pooling1d (Glob  (None, 128)               0
 alMaxPooling1D)

 dense (Dense)               (None, 10)                1290

 dense_1 (Dense)             (None, 1)                 11

=================================================================
Total params: 249129 (973.16 KB)
Trainable params: 249129 (973.16 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

1. Embedding Layer:

   An embedding layer was used with dimensions defined by the length of the word index obtained from the tokenizer, plus one. The output dimension was set to 100, and the input length was restricted to the maximum sequence length.

2. Convolutional Layer:

   A one-dimensional convolutional layer with 128 filters and a kernel size of 5 was implemented, utilizing the rectified linear unit (ReLU) activation function.

3. Global Max Pooling Layer:

   A global max-pooling layer was applied to extract the maximum value across the temporal dimension, enhancing feature extraction.

4. Dense Layers:

   A dense layer with 10 units and ReLU activation function was incorporated, followed by a final dense layer with a single unit and sigmoid activation for binary classification.

Additionally, we experimented with including an LSTM layer. However, due to limited data causing overfitting, we opted not to include this layer in the final architecture.

3. Skills Named Entity Recognition(NER):

- Noun phrases identified as skills by the classification model are selected for further processing.
- These identified skills are then fed into a Named Entity Recognition (NER) model.
- The NER model is designed to specifically recognize and extract skills from the input noun phrases.
- This additional step aims to enhance overall accuracy by focusing on the precise identification of skills within the classified data.
- The final output would be skills extracted from the resume

3. Job title Identification

- **Identify Job Designation using NER (Named Entity Recognition):** Utilize NER to extract job titles from the experience section of the resume.

- **BERT Embedding:** Apply BERT (Bidirectional Encoder Representations from Transformers) embedding to convert the extracted job titles into dense vectors. BERT embeddings capture the contextual meaning of words and phrases, which can enhance the performance of downstream tasks.
- **Mapping Alternate Designations with Predefined ONET Designations:**

```
Extracted: Senior Data Analyst -> Mapped: Data Scientists
Extracted: Project Engineer -> Mapped: Project Management Specialists
```

  - **Method 1 (Word2Vec Similarity):** Use Word2Vec to compute the similarity between alternate job titles and the predefined ONET database titles. Replace alternate titles with the ONET titles based on the highest similarity score.
  - **Method 2 (Training Embedding Layer):** Train an embedding layer on tokenized and padded words representing alternate job titles. This method learns embeddings specific to the job titles in the dataset, allowing for better representation of similarity between titles.
- **Extract Timeline of Job using Regex:** Once the standardized job titles are obtained, use regular expressions (regex) to extract the timeline information associated with each job. Regex patterns can be crafted to capture dates, durations, or other temporal information mentioned in the experience section of the resume.

```
{'2013-09-02': 'Project Management Specialists', '2017-04-02': 'Data Scientists'}
```

4. Text Summarization

At the end of the process, summarization is performed using the Facebook BART-Large-CNN model. BART (Bidirectional and Auto-Regressive Transformers) is a transformer-based model that excels at various natural language processing tasks, including text summarization. The "Large" variant of BART indicates a larger model size with more parameters, allowing for better performance.