HPCC Systems Internship

**Resume Analyzer**: An automated solution to the recruitment process using NLP++

Charvi Dave

Intern

Lexis Nexis Risk Solutions

10th March 2023

Mentors:

Umesh Mahind

Nandhini Velu

David De Hilster

# ACKNOWLEDGEMENT

I would like to extend my sincere thanks to David De Hilster for his invaluable suggestions and support. His guidance, encouragement, and mentorship has helped me reach this stage and inspired me in innumerable ways. I am grateful that I had the opportunity to work with him.

I express my sincere gratitude to Rahul Jain, Harsh Desai, Umesh Mahind and Nandhini Velu for their steady guidance and support throughout my project tenure. I am thankful to them for their constant inspiration, and I appreciate their guidance and help tremendously.

Table of Contents

1. [ABSTRACT 4](#_Toc303759305)
2. [Introduction 5](#_Toc303759305)

2.1. problem definition5

2.2. proposed solution [6](#_Toc303759310)

1. What is [NLP++? 7](#_Toc303759305)
2. [CONVERSION OF WORD/PDF/IMage resumes to text files](#_Toc303759308) 8

4.1. [Conversion of Word/PDF Resumes to Text 8](#_Toc303759309)

4.2. [Conversion of Image Resumes to Text using OCR 10](#_Toc303759310)

1. [How to install nlp++ 11](#_Toc303759311)
2. [how to use nlp++ 12](#_Toc303759312)
3. creating the [resume analyzer 14](#_Toc303759315)
4. conclusion  [36](#_Toc303759315)
5. [REFERENCES 36](#_Toc303759315)

# ABSTRACT

This report covers a Resume Analyzer created in NLP++. The project is about developing an analyzer for resumes.

Resume analysis involves a multi-step process of automated and manual resume reviews. The intent is to extract key information from resumes about job applicants in the most efficient manner possible.

Resume analysis ensures that candidates are strictly aligned with the job's requirements and hiring manager's expectations. It is the process of qualifying a candidate based on their education, experience, and other relevant information.

Hence there is a need for an analyzer which can automatically parse and extract relevant information from these resumes which makes it easier for the Talent Acquisition team to identify and narrow down the right talent.

In this project I have created an Analyzer/parser in the programming language, NLP++, which automatically parses and extract relevant information from resumes. It quickly reads through a resume and extracts and classifies each piece of information into predefined fields, such as Skills, Education, Work Experience, Name, Email ID, etc.

NLP++ works on the principle of multiple passes that elaborate a best-first parse tree. It is a “smart” language and can learn things like humans do. It stores information in Knowledge Bases. After the Analyzer extracts the information, it stores it in the Knowledge Base.

# INTRODUCTION

Resume Analyzer is the implementation of an approach to apply various techniques for analyzing the resumes a company receives and retrieving the main sections.

Word/PDF/Image Resumes are converted into Text Files in Python. We then use the Text File of the Resume as an input in NLP++. According to the formatting and words, we can write Rules and use Knowledge in NLP++ which will extract various sections of the Resume i.e. Skills, Work Experience, Email Id, etc.

We create an Analyzer which consists of multiple passes, where each pass of code performs a particular task or extracts a different piece of information from the resume. When the Analyzer is run, all the passes of code run on the file that is selected. The sections of the resume are extracted and stored in text files and a Knowledge Base. These output files consist of parameters like experience, skills, education, etc. These files are easier to view and use than the actual resume.

We are attempting to reduce efforts on the company’s side. The companies can adopt the system as a part of their recruitment process.

## Problem Definition

The Talent Acquisition Team receives thousands of resumes every month and they need to go through all resumes and shortlist candidates that match the job requirements and expectations.

The companies, their HR/Talent acquisition team does not have enough time to read thousands of resumes and choose the best resume according to their requirements. There is no standard format in which resumes are being sent. It is a tough and cumbersome task.

## Proposed Solution

Creating a Resume Analyzer in NLP++ with the following steps:

1. Word/PDF/Image Resumes are converted to text files using Python coding.
2. OCR will be implemented in Python to convert Resumes from the Image form to a text file.
3. Create a Resume Analyzer Sequence in NLP++ with several passes.

The Analyzer takes the text file of the Resume as an Input. It goes through the Resume, identifies, and classifies sections of the Resume i.e., Name, Skills, Education, Work Experience, etc. It extracts the information and stores it in a Knowledge Base.

1. The Output consists of the sections of the Resume we want to view along with the content.
2. We can also view the Knowledge Base later that consists of the Sections of the Resume and other information as well. It is easier and less time consuming to view the KB than it is to go through the actual resume.

## What is NLP++?

NLP++ is the only computer programming language exclusively dedicated for natural language processing. It was created by Amnon Meyers and David De Hilster in 1998. It allows programmers to capture and apply linguistic and world knowledge, emulating processes by which humans read and understand text. NLP++ combines bottom up, island-driven, recursive grammar, and other methods in a multi-pass architecture that operates on one parse tree. It works with a hierarchical knowledge base (KB), called Conceptual Grammar (CG), to dynamically build and use stored knowledge in analysing text. Applications range from simple syntactic processing to full natural language understanding.

NLP++ and CG deploy with an Integrated Development Environment (IDE) called VisualText, which supports rapid development of text analyzers. VisualText is a developer's environment that exploits NLP++ and CG to rapidly elaborate text analyzers. Passes and KBs from one analyzer may be exploited to more rapidly construct and tailor new text analyzers.

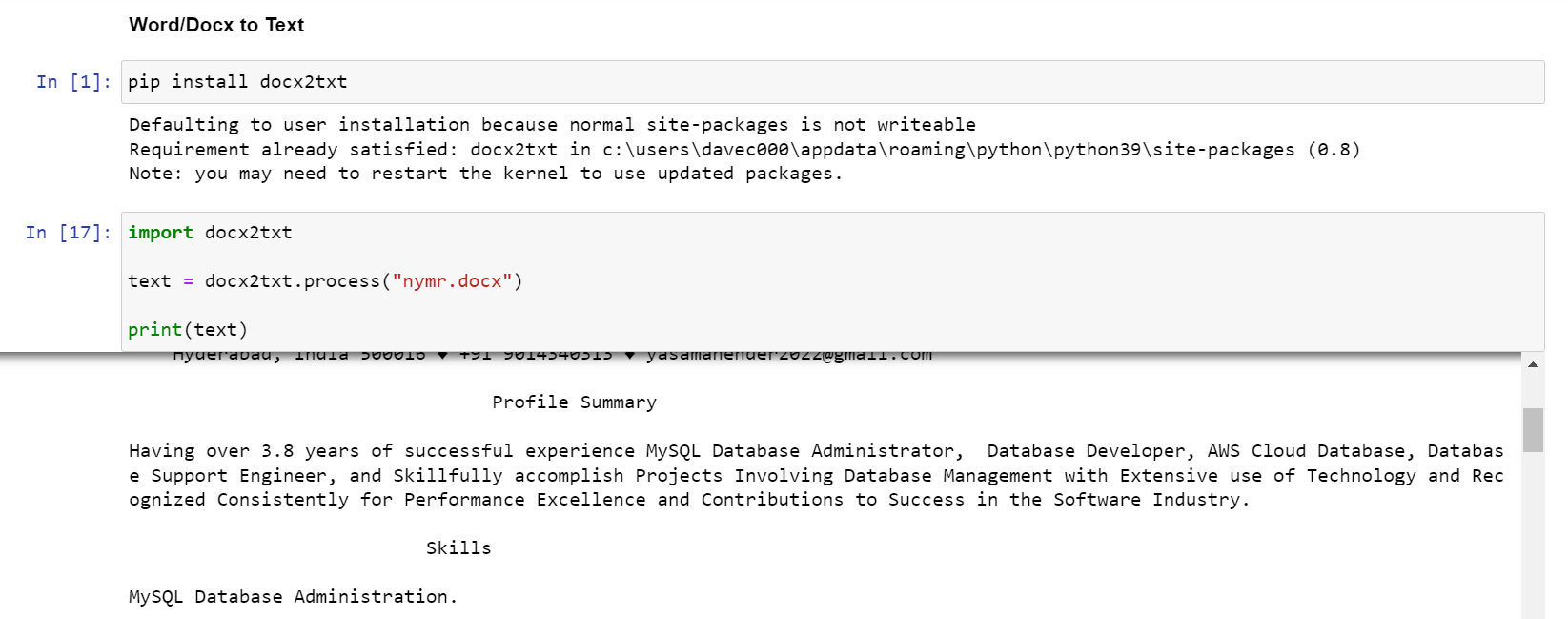
NLP++ takes text, breaks it down into tokens, builds up those tokens into syntactic trees, and builds and uses knowledge stored in Conceptual Grammar. The language includes functions, rules, operators, and variables specific to its internal representations of text and knowledge. NLP++ comprises general C or C++-like programming language constructs, as well as integrally addressing rule matches and the associated knowledge base.

For the two decades, the technology was privately owned and was licensed by private companies to process medial, social media, historical documents, and real estate text. In December of 2018, NLP++ and VisualText went open source.

**Note:** For security purposes, I have modified the Name, Phone Number and Email ID in the Resumes used.

# Conversion of Word/PDF Resumes to text Files

## Word To Text

****

## PDF To Text

**Graphical user interface, text, application, email

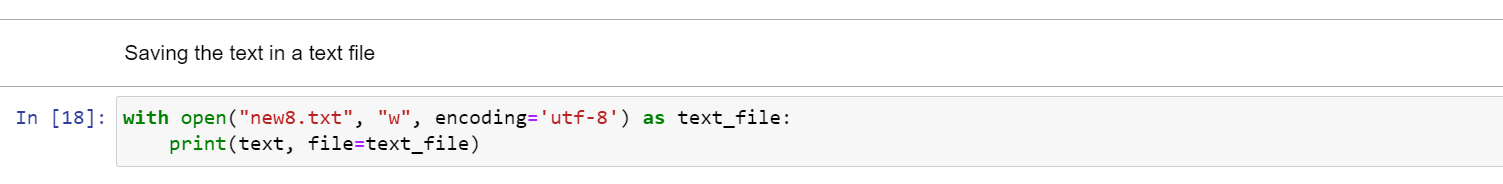
Description automatically generated**

**Graphical user interface, text, application, email

Description automatically generated**

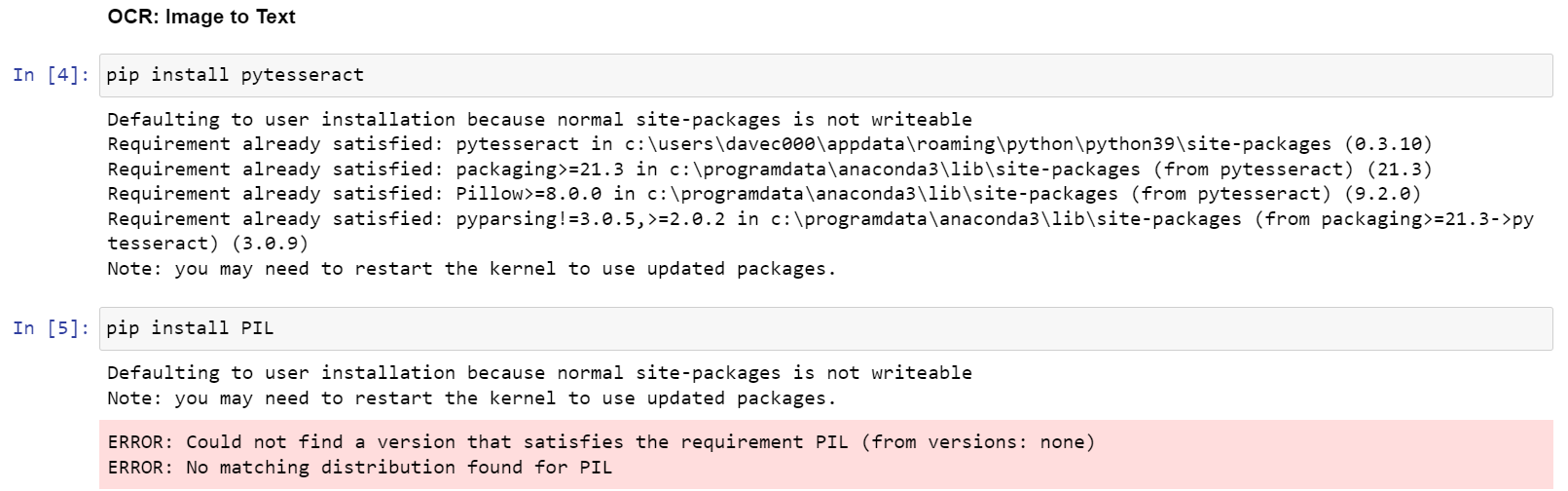


The Name, Phone Number, LinkedIn and Email ID has been erased here. We use this Resume for the explanation of the entire Analyzer later.

****



## Using OCR To Convert Image Resumes To Text

****

**Graphical user interface, text, application, email

Description automatically generated**

# HOW TO INSTALL NLP++



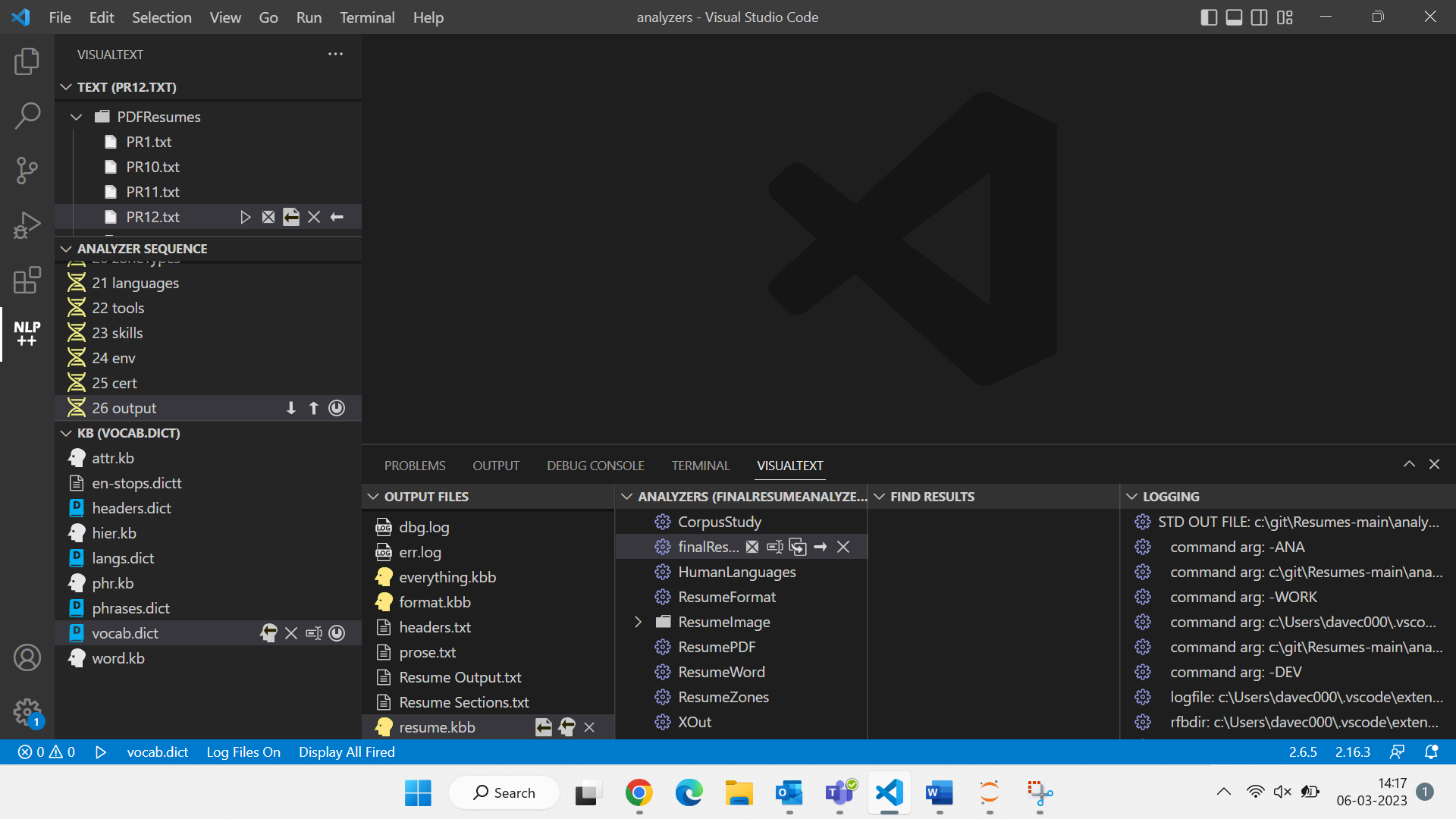
1. Install and open Visual Studio Code
2. Go to “Extensions” and install the NLP++ Extension.

A screenshot of a computer

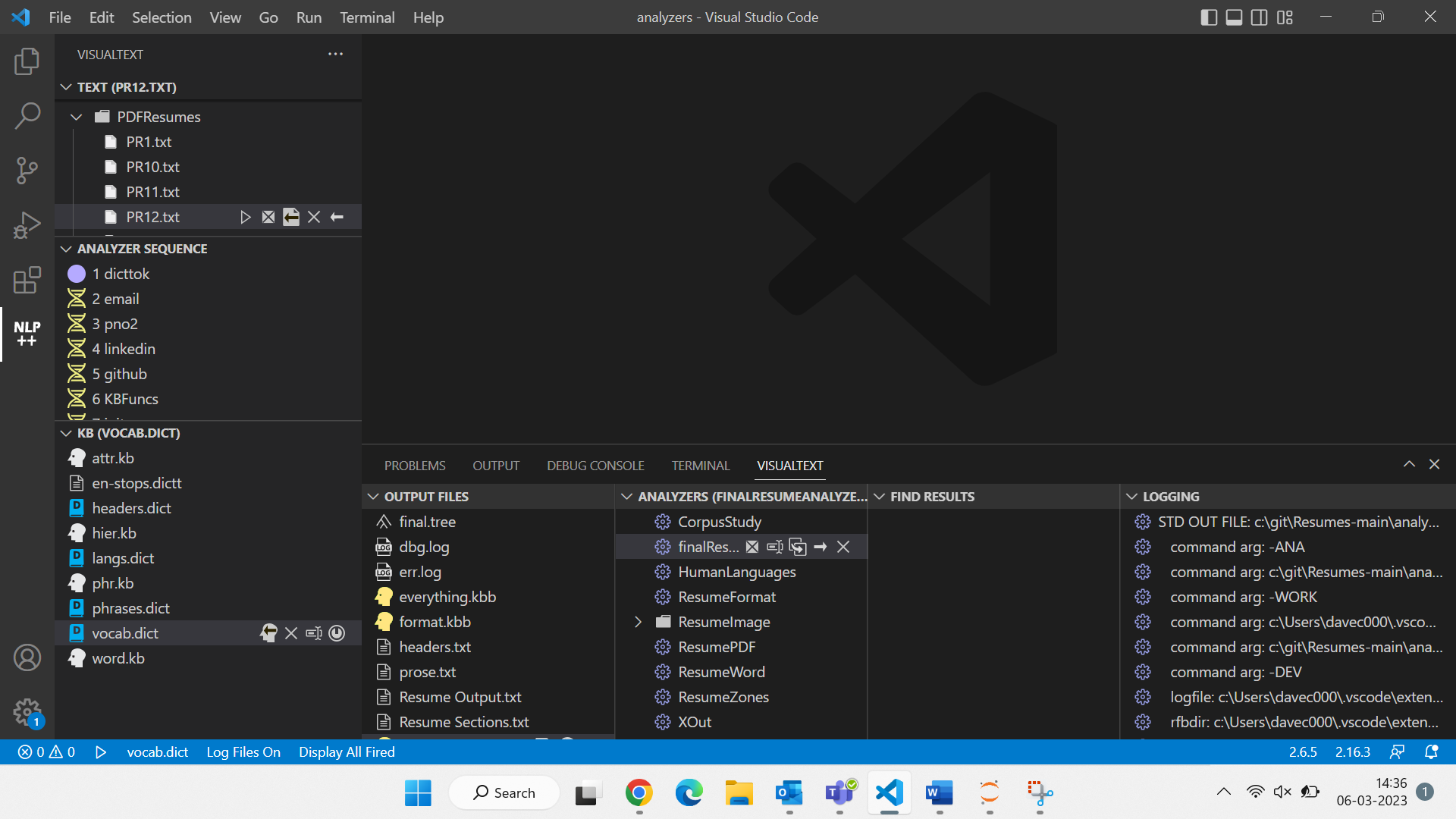
Description automatically generated

For further help on this: <https://www.youtube.com/watch?v=0U1VJxko4Jk>

1. Click on the NLP++ Icon
2. Ctrl+J bring forth the Ribbon at the bottom, Visual Text is the section of the Ribbon that we keep open and use.



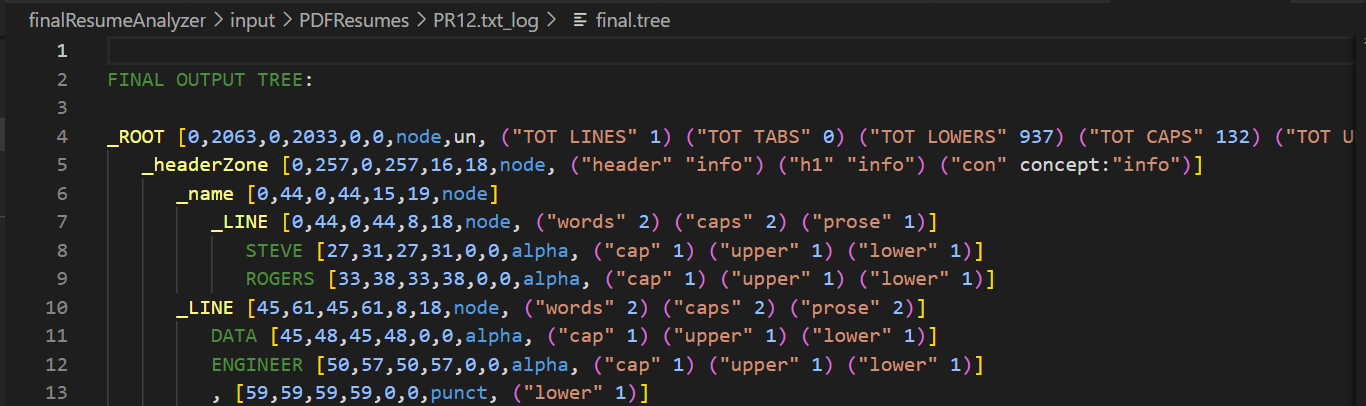
HOW TO USE NLP++



1. First upload the Resume(s) in the top left “text” corner.
2. The Analyzer Sequence consists of multiple passes which we have created, each pass is an NLP++ code that performs a particular function.
3. “KB” stands for Knowledge Base and consists of the KBs and dictionary files that we have created.
4. “Output Files” consists of all the Output Files that are available after running the code. It consists of the final tree and log files by default and the text files that we have created in our passes.
5. “Analyzers” consists of all the Analyzers we have created.

finalResumeAnalyzer is the Analyzer that we have created and used.

The Final Tree Output File:



When you click on the  “run” button on the text file, it runs the entire Analyzer Sequence, all the passes, on that text file.

Graphical user interface, text, application

Description automatically generated

CREATING THE RESUME ANALYZER

This is the Resume we are working on:

Text

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

A picture containing background pattern

Description automatically generated

## Steps:

1. Click on the Analyzer to use it.

Graphical user interface, text

Description automatically generated

1. Add a new pass for every new function that needs to be performed.

Text

Description automatically generated with medium confidence

1. Create 2 dictionaries in the bottom left “KB” section – One Dictionary, Headers should consist of a list of all possible “Headers” (Skills, Education, Experience, etc) and One Dictionary, Vocab should consist of a list of various tools, languages, skills, technologies that the Analyzer should detect.

A screenshot of a computer

Description automatically generated with low confidence

1. The first pass is dicttok which is the built-in English dictionary tokenizer.

Text

Description automatically generated

1. The second pass is “email”.

Here I have written the NLP++ code to extract the email ID of the person from their Resume.Text

Description automatically generated

@NODES \_ROOT means to start looking at the Root of the Tree.

@RULES region is where we define the Rules, the condition to match to extract the Email Id.

1. \_xWILD means the text can be any data, “plus match” means it can match more than one letter, number, underscore, dot and dash sign.
2. \@ means after the \_xWILD, there has to be an @ sign.
3. \_Xalpha refers to any letters after the @ (mostly gmail, yahoo, etc)
4. \. Refers to a full stop
5. \_xWILD again means the text can be any data, “plus match” means it can match more than one letter, number, underscore, dot and dash sign. (e.g. @edu.in)

@POST region is where we have created a text file “Resume Output”. Here we are storing the Email ID that we have extracted in the Rules region.

Graphical user interface, text, application

Description automatically generated

* this is the button to highlight the text that we have extracted from the file.

E.g. when I click on that, I get this



Resume Output txt file that we created:

Graphical user interface, text, application

Description automatically generated

1. The next pass is “pno2” in which we extract the Phone Number.



@RULES

I have defined the Rule to extract the Phone Number. We are working with Indian Resumes hence the phone numbers would be 10-digit numbers in the form of 9837389976, +91-9837389976, +919837389976 or (+91) 9837389976.

The Brackets and “+91” are kept as “optional” since it is not necessary one would use those. What is compulsory is the 10-digit phone number as mentioned in (4). Trig means it is a compulsory match.

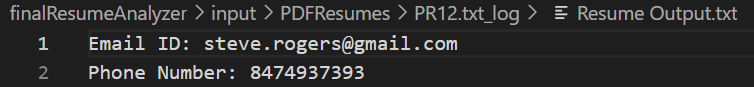
@POST

Here in the first length, we are storing the length of the string, only the last part (4). Sometimes, when the phone number is written as +919837389976, the entire thing is considered as the number hence its length would be 12. In the first if statement, if the number’s length is 12, we are using strpiece (substring) and extracting the 10-digit actual phone number, excluding the “91” in front.

In the else if statement, if the number is 10 digits long then we are automatically extracting and printing that into our “Resume Output.txt” file.

This is what we get when we Highlight the text. ()



This is what we get after running the code:

1. The next pass is “linkedin”.

Here we are extracting the LinkedIn Account/Handle of the Person from their Resume.

Text

Description automatically generated

@RULES

This is the basic format that is there for one’s LinkedIn account link: linkedin.com/in/accounthandle

What we are doing here is extracting their account handle. The format is fixed, but their account handle (7) can consist of anything, multiple letters, numbers, underscores, etc hence it is mentioned as “plus match”.

@POST

Here we are just printing the LinkedIn Account Handle into the “Resume Output.txt” file.

When we Highlight the Text:



This is what we get after running the code:Text

Description automatically generated

1. The next pass is “github”.

Here we are extracting the Github Account/ Handle of the Person from their Resume.



@RULES

Basic format for Github Account: github.com/accounthandle

What we are doing here is extracting their account handle. The format is fixed, but their account handle (5) can consist of anything, multiple letters, numbers, underscores, etc hence it is mentioned as “plus match”.

@POST

Here we are just printing the Github Account Handle into the “Resume Output.txt” file.

This Resume that we have used does not include a Github Account Link hence there are no screenshots, but it is similar to the “Linkedin” pass.

1. The next pass is “KBFuncs”.

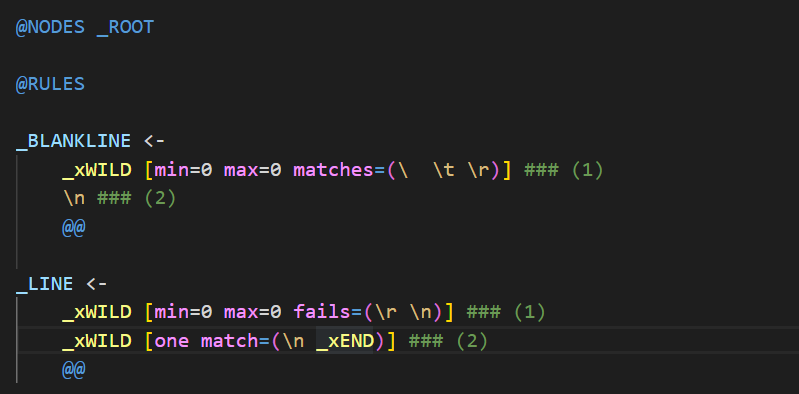
This pass just consists of Knowledge Base (KB) - related functions. e.g. Functions to Add Concepts, Attributes, Attribute Values to the KB, to display the KB, to save the KB, etc.

1. The next pass is “init”.

Here we are retrieving the Concept of the Format of the Resume and the Resume.

1. The next pass is “Lines”.

Here we are just defining what should be considered/identified as a Line and what should be considered as a Blank Line.



@RULES

\_blankline

Any blank space such as a tab \t, a space \ ,etc followed by a new line \n is considered to be a blank line.

\_line

1. Denoted \_xWILD i.e. any letter, number, special characters, etc are considered to ba apart of a line. It fails when it meets a blank line.
2. When a line is followed by a new line, or \_xEND (the end of the document), then it is recognized as a line.
3. The next pass is “bullets”.

Here we are searching for the bullet points in the document. It can recognize all different types of bullet points.

Text

Description automatically generated

@PRE

This Region is checked first, before checking the Rule.

<2,2> refers to (2) in the @RULES Region. Length is set as 1 which denoted the length of the bullet point since a bullet point is just one single bullet.

@RULES

1. \_xSTART refers to the Start of the Document
2. In \_xWILD, the “one match” means it should match either a \_xCTRL (ANSI characters) or \_xPUNCT (punctuation). Each of these indicate a bullet point. It should fail when it encounters a bracket.

@POST

L refers to a local variable, G refers to a global variable, N stands for a Node variable.

Here we are just storing the Concept of Bullet points in the Format Concept.

When we Highlight the Text:

Text

Description automatically generated

1. The next pass is “RemoveWhiteSpace”.

The White Spaces between words/lines/paragraphs are irrelevant for us now and we only want to work with the words, so we are removing all the White Spaces.

Graphical user interface, application

Description automatically generated with medium confidence

@RULES

\_xWHITE denotes a White Space.

@POST

Here we are excising/removing the White Space.

(1,1) refers to the node (1) in the RULES Region.

1. The next pass is “wordTypes”.

Here we are identifying different types of words, which words have Upper Cases and Lower Cases in them, what should we consider as a Prose, what should we consider as a Header.

Text

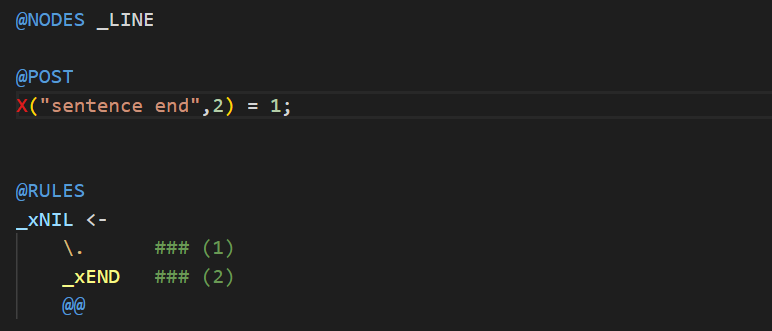
Description automatically generated

@RULES

\_xALPHA refers to any word.

1. The next pass is “sentEnd”.

Here we are searching for the end of each sentence.



@RULES

A full stop or the End of the Document indicates the end of a sentence.

When we Highlight the Text:

Text

Description automatically generated

1. The next pass is “Prose”.

Prose refers to a paragraph or a group of sentences/text all together.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

When we Highlight the Text:

Text

Description automatically generated

1. The next pass is “Headers”.

Here we are searching for “Headers” in tbe Resume. Headers consist of the sections of a Resume i.e. Skills, Education, Work Experience, etc. Our Main Objective with this Project is to extract Headers and the sections/information within each Header in the Resume.

Text

Description automatically generated

This is the Headers dictionary that we have created earlier, it consists of a list if all possible Headers of a Resume.

Text

Description automatically generated

@RULES

The Header is essentially the Line (one word or multiple words) before the information within the Header.

@POST

We are considering two types of Headers, level 1, and level 2.

If the first “if” statement follows, the Header is a Level 1 Header. E.g. SKILLS

If the Word(s) is present in the headers.dict file and if all the letters of the words(s) are in all Caps , then it is a Level 1 Header.

If the “else if” statement follows, the Header is a Level 2 Header. E.g. Skills

If the Word(s) is present in the headers.dict file and if the first letter of each Header word is in Upper Case and the rest are in Lower Case, then it is a Level 2 Header.

We are printing all the Headers into out Headers.txt file.

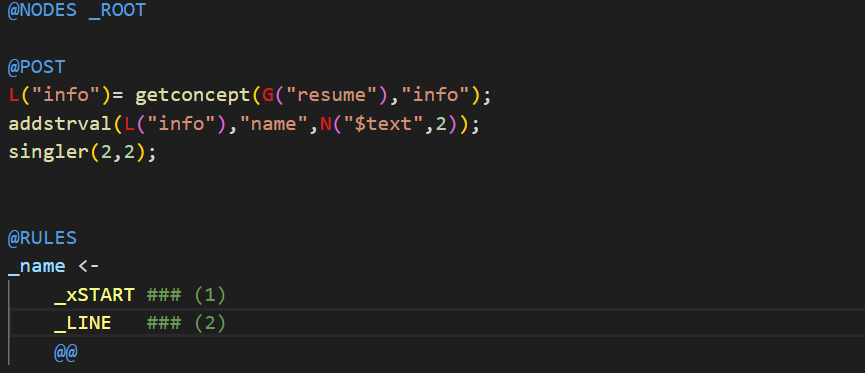
When we Highlight the Text:

Text

Description automatically generated

1. The next pass is “pname”.

Here we are extracting the person’s Name and storing it into the KB.



@RULES

Here we are considering the First Line of the Document to be the Person’s Name.

@POST

Here we are storing the Person’s Name in the “info” section of the Resume. The first section of the Resume which has no header and consists of the Person’s personal information is called the “info” section.

When we Highlight the Text:

1. The next pass is “info”.

Here we are extracting the “info” section of the Resume. It is the first section of the Resume which has no header and consists of the Person’s personal information,



@RULES

We start at the Start of the Document and extract all the information (letters, numbers, special characters, etc) till we encounter a Header. We extract all the information from the Start till the First Header.

@POST

Here we are calling this section as “info” and storing it in the Headers text file.

When we Highlight the text:

Text

Description automatically generated

Headers.txt file:

Text

Description automatically generated

1. The next pass is “proseOut”.

Here, we are just printing out one of our previous passes, “prose”.

Text

Description automatically generated

@RULES

We are extracting the Prose that we defined in an earlier pass.

@POST

We are printing the Prose into the Prose.txt file.

When we Highlight the text:

Text

Description automatically generated

Prose.txt file:

Text

Description automatically generated with medium confidence

1. The next pass is “headerZone”.

Here we are extracting the section/information within each Header of the Resume.

Text

Description automatically generated

@RULES

We start at a Header and stop extracting once we encounter the next Header. We are essentially extracting the information between two Headers.

When we Highlight the Text:Text

Description automatically generated

1. The next pass is “zones”.

Here we’re just printing out the Header Zones/Sections that we extracted in the previous pass, “headerZones” and we’re storing the Concept of the Zones in the KB.



@RULES

Here we are extracting the Header Zones using the previous pass.

@POST

We are storing the Concept of each Header and Zone in the KB.

We are also printing the Header Zones into the “Resume Sections.txt” file.

When we Highlight the Text:Text

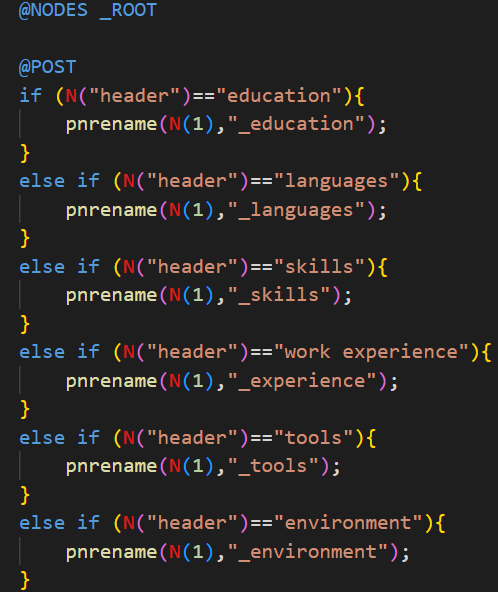
Description automatically generated

“Resume Sections.txt” file:

Text

Description automatically generated with low confidence

1. The next pass is “zoneTypes”.

Text

Description automatically generated

In the Final Tree, each Header Zone is normally indicated as \_headerZone, but here we are replacing that with an underscore following the name of the Header.

For e.g. for skills, \_headerZone is renames to \_skills.

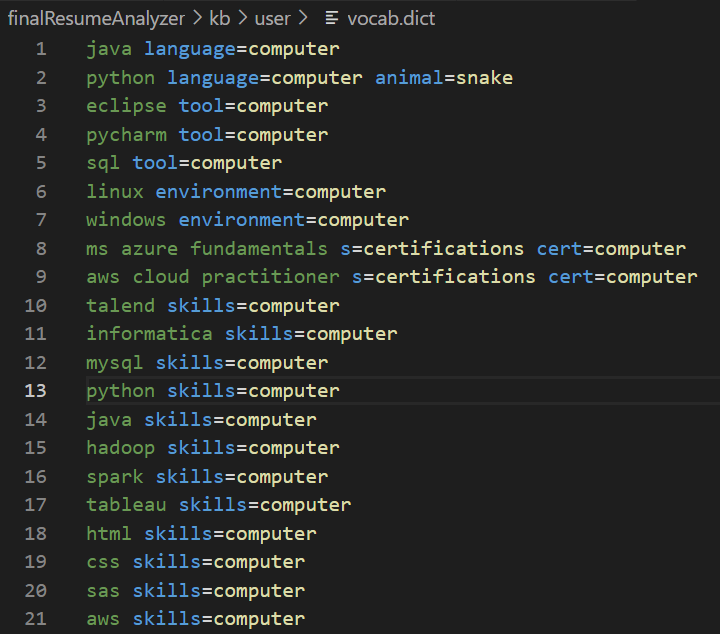
Final Tree:

Text

Description automatically generated

1. The next 5 passes are very similar to each other. They are “languages”, “tools”, “skills”, “env”, “cert”.

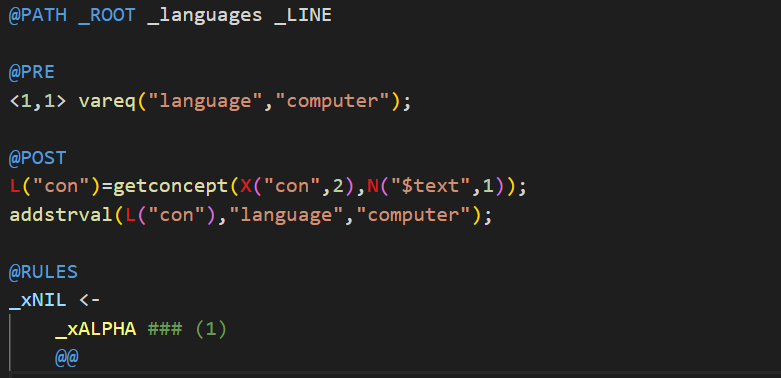
This is the vocab.dict file that we created at the start.



Here we have listed a bunch of skills, technologies, software, environments, certifications, etc. We can add a lot more to this list.

In these next 5 passes we are just searching too see if we have the languages, tools, skills, environments and certifications in this vocab.dict. If it is present, then it will print it out and store it in the KB.

“languages” pass



When I highlight the text:

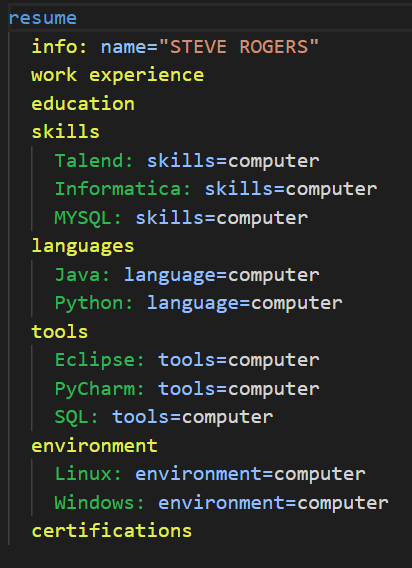
A picture containing graphical user interface

Description automatically generated

The rest of the passes are exactly the same, except “language” is replaced by “tools”, “skills”, “environment” and “certifications” respectively.

Here is the Final KB, where we have extracted and stored the Knowledge from different Sections.

Resume.kbb



1. The last pass is “output”.

Here we are just saving all of the final Knowledge Bases.

Text

Description automatically generated

format.kbb

Text

Description automatically generated

1. These are all the passes in our Resume Analyzer Sequence.

# 

# CONCLUSION

NLP++ is a powerful tool to work on Natural Language. It thinks like a human, stores information in a Knowledge Base and learns from it. Creating a Resume Analyzer in NLP++ is a revolutionary thing to do. The project can further be extended as NLP++ can help us explore and learn much more. Through this solution we are attempting to automate and simplify the recruitment process. Time and effort taken for this process is reduced.

REFERENCES

1. [NLP++ – VisualText](http://visualtext.org/nlp/)
2. [NLP++ - Natural Philosophy Wiki](https://wiki.naturalphilosophy.org/index.php?title=NLP%2B%2B)
3. <https://www.youtube.com/@nlpvisualtext3212/videos>
4. <http://www.textanalysis.com/>
5. [Link to the YouTube video where I discuss the Analyzer with David De Hilster](https://www.youtube.com/watch?v=Xt1yeIl1ySM&list=PLrRF6oTCn26CGezjYzwqkE2AaoTtb5wHM&index=15)