Resume Parser

1. Introduction

In today's digital age, companies and recruiters receive a large number of resumes in various formats, making it challenging to extract and analyse the information efficiently. The Resume Parser project aims to develop a solution that automates the extraction and organization of essential information from resumes, such as names, email addresses, phone numbers, and skills. By leveraging natural language processing (NLP) techniques and cloud services, the project will provide an efficient and accurate method for processing resumes and storing the extracted data.

2. Objectives

The main objectives of the Resume Parser project are as follows:

- Develop a system capable of receiving resume files in PDF or DOCX formats.

- Store the received resume files in an S3 bucket for processing and storage.

- Implement a Lambda function to convert PDF and DOCX files into plain text format.

- Utilize the pre-trained spaCy model for NLP to extract key information, including names, email addresses, and phone numbers from the plain text resumes.

- Store the extracted information in a structured and efficient manner.

- Optional: Extract additional details such as skills from the resume text.

- Automate the resume parsing process by setting up S3 event notifications for real-time processing.

3. Methodology

The project will be executed according to the following steps:

1. Research NLP techniques: Gain a solid understanding of NLP and its applications in resume parsing to ensure effective implementation.

2. Learn S3 bucket management: Acquire knowledge on creating and managing S3 buckets to store and retrieve resume files.

3. Store test data in S3 bucket: Utilize publicly available test data, such as the Resume Dataset on Kaggle, to populate the S3 bucket and verify successful storage.

4. Research Lambda function execution: Explore the process of running Lambda functions using tools like localstack to ensure seamless conversion of resume files into plain text format.

5. Extract text from resume files: Investigate techniques for extracting text from PDF and DOCX files, such as utilizing PDFMiner and doc2text libraries.

6. Create Lambda function for conversion: Develop a Lambda function that converts PDF and DOCX files to plain text or HTML, making use of the researched extraction techniques.

7. Set up S3 event notifications: Configure event notifications on the S3 bucket to trigger the resume parsing process automatically upon file upload.

8. Tokenize and parse data with spaCy: Utilize the pre-trained spaCy model to tokenize and parse the plain text resumes, extracting information like names, email addresses, and phone numbers.

9. Store parsed information: Design an efficient data storage system to store the extracted information, ensuring easy retrieval and further analysis if needed.

10. Optional: Extract additional details: Research and implement techniques, such as regex matching, to extract skills or other relevant information from the resume text.

11. Test and validation: Validate the accuracy and efficiency of the resume parser by processing a variety of resumes and comparing the extracted information with the original documents.

12. Documentation and optional enhancements: Document the project's implementation, including the chosen techniques and any optional enhancements made during the development process.

4. Potential Challenges

During the development of the Resume Parser project, some challenges may arise:

1. Incomplete information: Resumes may not provide all the required information, leading to missing data points in the parsing process.

2. Inconsistent formatting: Resumes may have varying formats, styles, and layouts, making it challenging to extract information consistently.

3. Language variations: Resumes in different languages may require additional language-specific processing techniques to extract the desired information accurately.

4. Incorrect labelling: The pre-trained model or extraction techniques may misclassify or mislabel certain information, leading to inaccurate parsing results.

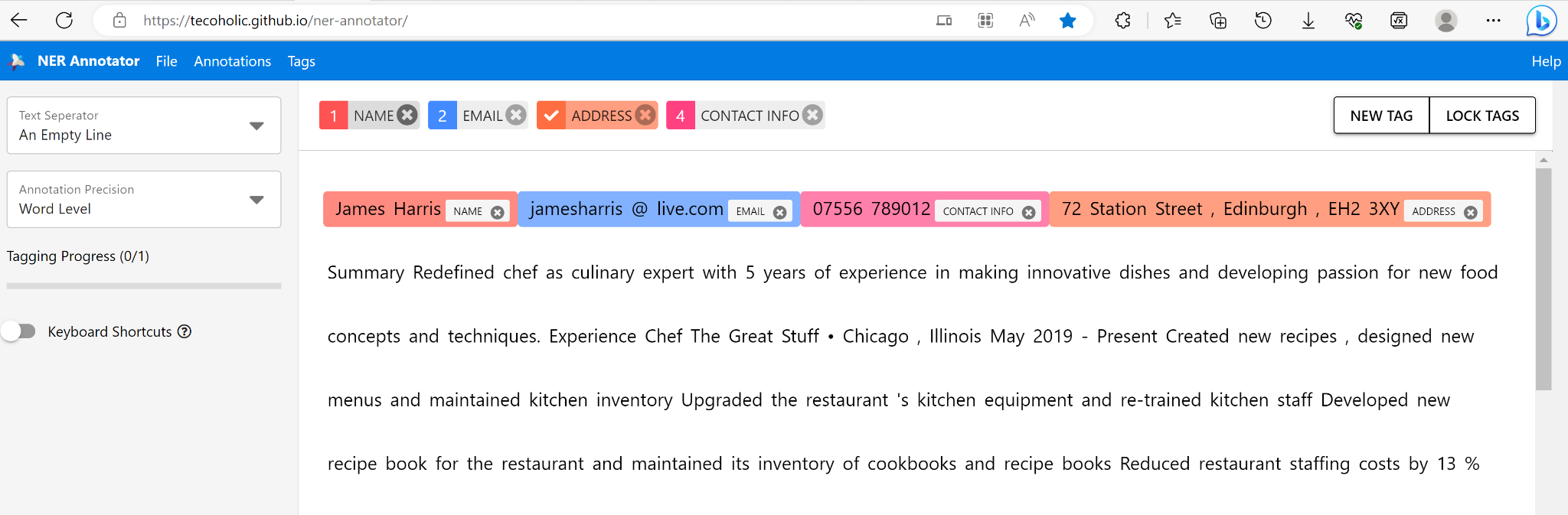
5. Workflow  
  
The process of training the model requires 4 steps:

1.Preparing the data -

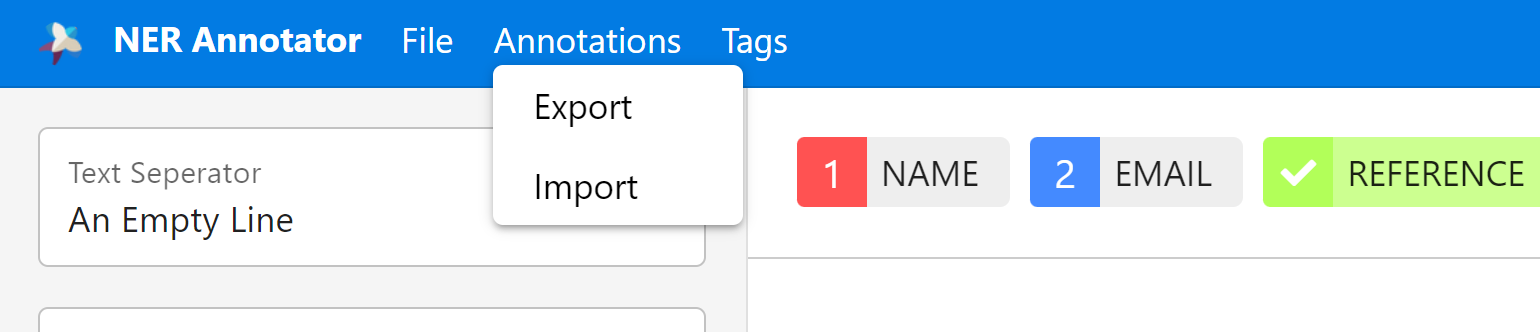
* Finding CV’s/Resumes that we can use as templates and converting to TXT files to allow us to edit.
* Formatting the Data. Adding any necessary components that we want to extract if not already included in the template.

2.annotating the data -

* Using [NER Annotator for SpaCy (tecoholic.github.io)](https://tecoholic.github.io/ner-annotator/) a free NER annotation tool to add and label the corresponding entities within the TXT files
* Once all the annotations have been exported, we separate the Json annotations into two folders. one for **Evaluative data(**Data Ai Model uses to analyse performance/rate of learning etc.) and one for **Training Data(**Data Ai Model is trained on)

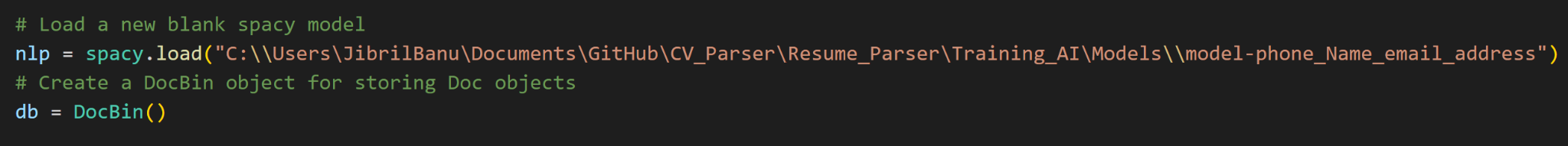


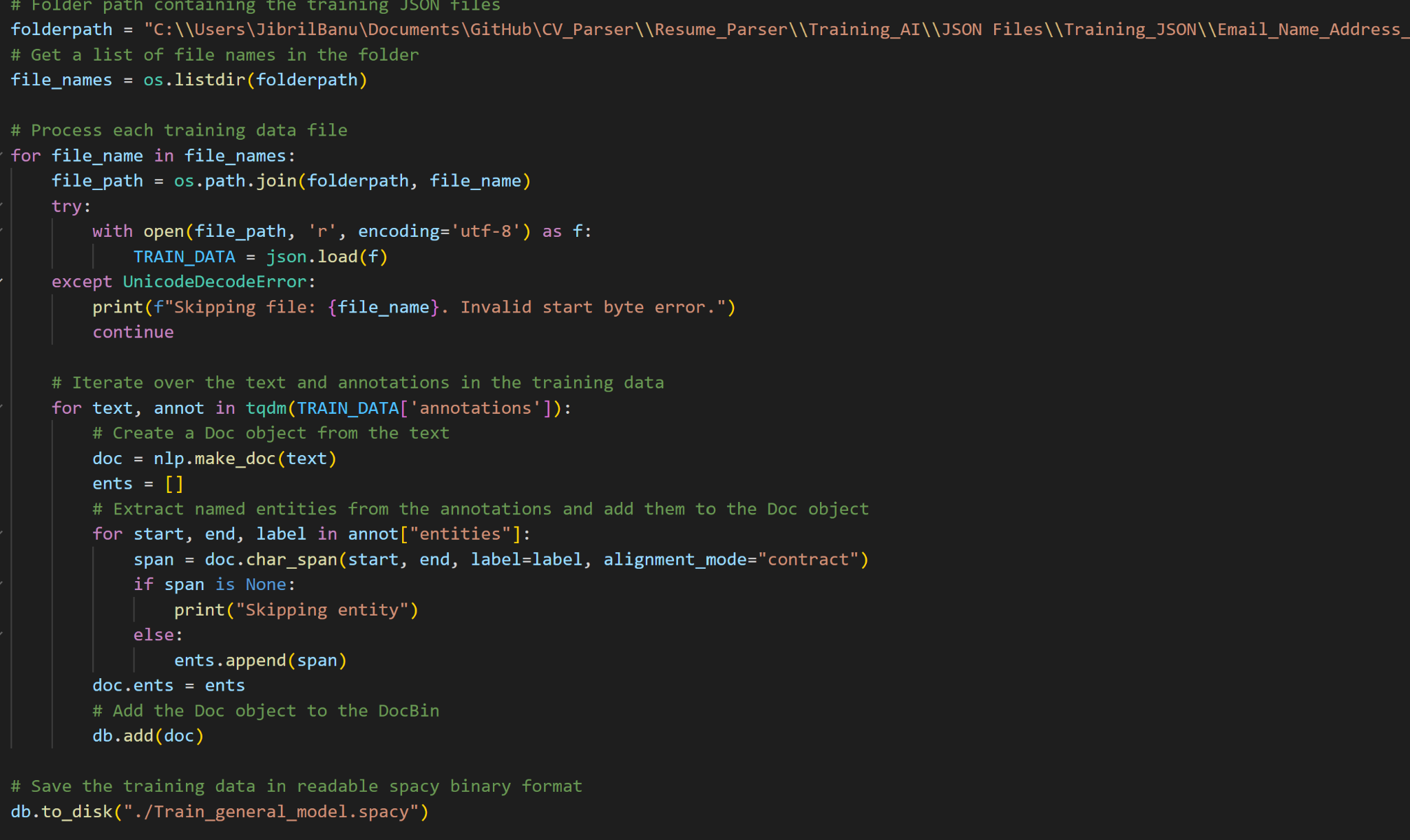
* Exporting the annotations as JSON files using the annotations widget.



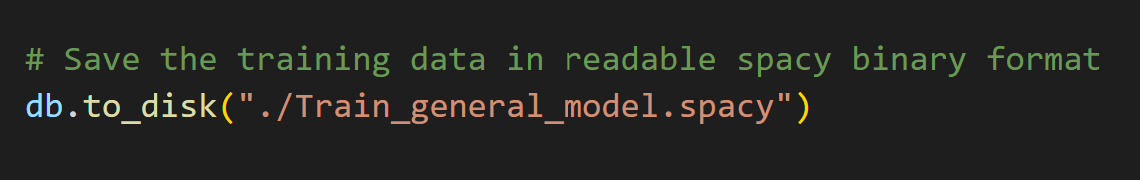
3.converting the data -

Once the data has been annotated, we then need to convert it into spacy’s binary format.

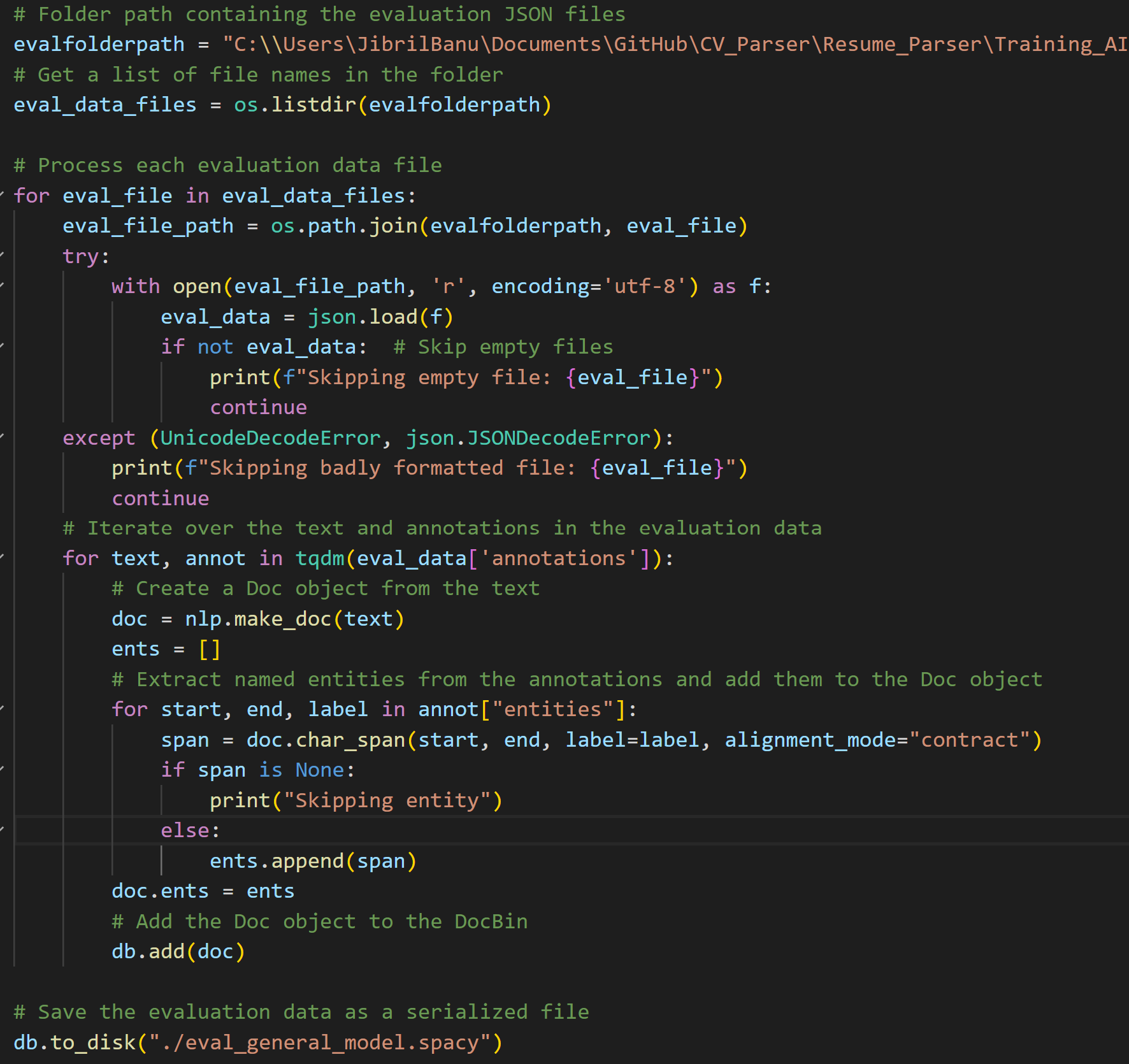
This is a requirement for training using Spacy. We do this by saving the Data in a DocBin object. Here you can see us initialise the Docbin object as well as load one of our trained models.

We are training our model iteratively so we are continuously updating the model we are using.

Above you can see we are reading the files from a folder, iterating over the text in the data and finally saving the training data in a readable spacy binary format using the db.to\_disk method.



We repeat the process with the evaluative data in a separate DocBin object.



Once we run this code, there will now be two formatted binary data files ready to use for training.

4.Training

For the training to start, we must have a config.cfg file ready.

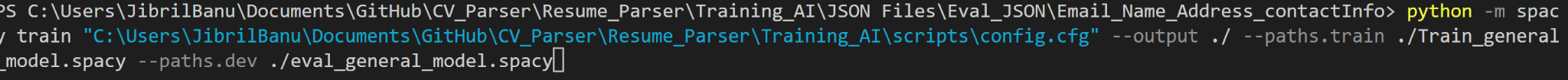
The config file contains a lot of the instructions for the training.

If you have no config file, you can get it by running this in the terminal:

python -m spacy init config config.cfg --lang en --pipeline ner --optimize accuracy

We then run the cli commands:

python -m spacy train "C:\Users\JibrilBanu\Documents\GitHub\CV\_Parser\Resume\_Parser\Training\_AI\scripts\config.cfg" --output ./ --paths.train ./Train\_general\_model.spacy --paths.dev ./eval\_general\_model.spacy



This References our config file, and the two binary data files.

6. Conclusion

The Resume Parser project aims to streamline the resume processing and information extraction process using NLP techniques and cloud services. By automating the conversion of resume files into plain text format, extracting key information using spaCy, and storing the parsed data efficiently, the project will provide an effective solution for recruiters and companies to manage and analyse resumes more effectively.