Overview: The system requires to take in a CV, and output a qualification rating for the O-1A visa between ‘low’, ‘medium’, ‘high’ which represents the likelihood that the candidate meets the O-1A standard. I used a FastAPI endpoint to accept a CV file upload, extract text, compute a score and return a JSON response with the evidentiality items and overall qualification rating.

Preprocessing: In my script I preprocess the pdf document to extract text and remove stopwords and normalized text so that each input to my model is a string word.

Extracting key information: This task involves Name Entity Recognition(NER) with a lot of word matching involved, such as trying to assess how much the candidate meets the 8 criterias: awards, membership in associations, press, judging, original contributions, scholarly articles, critical employment, and high salary. For NER tasks I decided to use a pre-trained transformer language model because they are trained on very large datasets, yielding significant accuracy in word recognition. I decided to use a Bert model as it is proven to be efficient in word matching and NER recognition, and I also have past experience fine-tuning Bert. For the pre-trained model, I used the “bert-large-cased-fine tuned-conll03-english” model because this model has been fine-tuned on the CoNLL-2003 dataset and is widely referenced in tutorials and production projects for its performance on standard NER tasks.

- Can be further improved by fine-tuning the model, I have included a snippet of the fine-tuning code and the csv dataset.

Evaluation: I pass the pre-processed text to the model and Bert generates a dictionary of extracted entity labels, such as ‘B-PER’, ‘B-ORG’, ‘B-MISC’, along with the extracted word. Based on the extracted entity labels, I matched the word with commonly associated keywords, and if there is a match, I keep track of it. After parsing the file, I check how many of the criteria the applicant matches, and currently I am assuming all the criterias have the same weighting, so i count how many criterias does the applicant meet and determine a score;

- ‘Low’ if applicant meets < 4, so less than a majority of the criterias
- ‘Medium’ if more than majority
- ‘High’ if more than 6 which is >= 75%

Further improvements:

1. Currently I’m just using a pre-trained model, but the accuracy can vary as it is very generalized. We can pre-train the model by creating a dataset with keywords that past successful applicants have used, so we are sure that if there is a match, the candidate has a higher chance of being accepted. Additionally:
    a. Currently I am just using 1 or 2 keywords, but can further enhance accuracy by using a separate model for name matching, perhaps an e5 model.
2. All the criteria currently I am using are weighted the same, but I’m sure some criterias are weighted more than others, if we have access from past visa applications I want to filter and see out of the 8 criterias, which criteria are favored the most, ie: are applicants with academic awards are more likely to be accepted? I want to use a separate transformer model to generate weighting for all 8 criterias, with input as past visa applications, and the output as whether the application was accepted or not.

3. Currently I am using the evaluation criteria as how many criterias the applicant meets, but I want to further enhance this by researching state-of-the art evaluation criterias, as I am sure there are many other evaluation criterias that are more accurate.
4. Can add further filtering for input, such as PDF, Docx and others for a better adaptability to the end user. Also can add error handling and scalability, maybe deploying the training API on GPU cloud for load balancing.