In this notebook I will be using SPaCy to create model which will extract main points from a resume. We will train the model on almost 200 resumes.

After the model is ready, we will extract the text from a new resume and pass it to the model to get the summary.

Importing the libraries

In [ ]:
```
import spacy
import pickle
import random
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

Losding the data

In [ ]:
```
train_data = pickle.load(open('/kaggle/input/resume-data/train_data.
```

In [ ]:
```
# Let's see the data
train_data[0]
```

Training the data on the model

We will first load a black SpaCy english model. Then we will write a function which will take the training data as the input. In the function, first we will add a ner i.e. Named Entity Recognition in the last position in the pipeline. Then we will add our custom labels in the pipeline.

In [ ]:
```
# Loading the blank SpaCy english model
nlp = spacy.load('en')

```

In [ ]:
```
nlp = spacy.blank('en')

def train_model(train_data):
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')
        nlp.add_pipe(ner, last = True)

    for _, annotation in train_data:
        for ent in annotation['entities']:
            ner.add_label(ent[2])


    other_pipes = [pipe for pipe in nlp.pipe_names if pipe != 'ner']
    with nlp.disable_pipes(*other_pipes):  # only train NER
        optimizer = nlp.begin_training()
        for itn in range(10):
            print("Starting iteration " + str(itn))
            random.shuffle(train_data)
            losses = {}
            index = 0
            for text, annotations in train_data:
                try:
                    nlp.update(
                        [text],  # batch of texts
                        [annotations],  # batch of annotations
                        drop=0.2,  # dropout - make it harder to mem
                        sgd=optimizer,  # callable to update weights
                        losses=losses)
                except Exception as e:
                    pass

            print(losses)
```

In [ ]:
```
# Let's train the model
train_model(train_data)
```

In [ ]:
```
# Let's save the model for further use
nlp.to_disk('nlp_model')
```

Model testing

Let's check out how our model is performing. For this we will pass a new resume to this model.

In [ ]:
```
# Installing PyMuPDF for getting the text data from the resume pdf
!pip install PyMuPDF
```

In [ ]:

```
import sys, fitz

fname = '/kaggle/input/resume-data/Alice Clark CV.pdf'
doc = fitz.open(fname)
text = ""
for page in doc:
    text = text + str(page.getText())

tx = " ".join(text.split('\n'))  # for removing the next line charac
print(tx)
```

In [ ]:

```
# Now we will pass this extracted text to our model
nlp_model = nlp.from_disk('/kaggle/input/resume-data/nlp_model/')

doc = nlp_model(tx)
for ent in doc.ents:
    print(f'{ent.label_.upper():{30}}- {ent.text}')
```

This is amazing. But we can make this model more accurate by training it on more data.

And I will be constantly updating this notebook to make it better.