Name: Alexis Collier

Email: info@alexiscollier.com

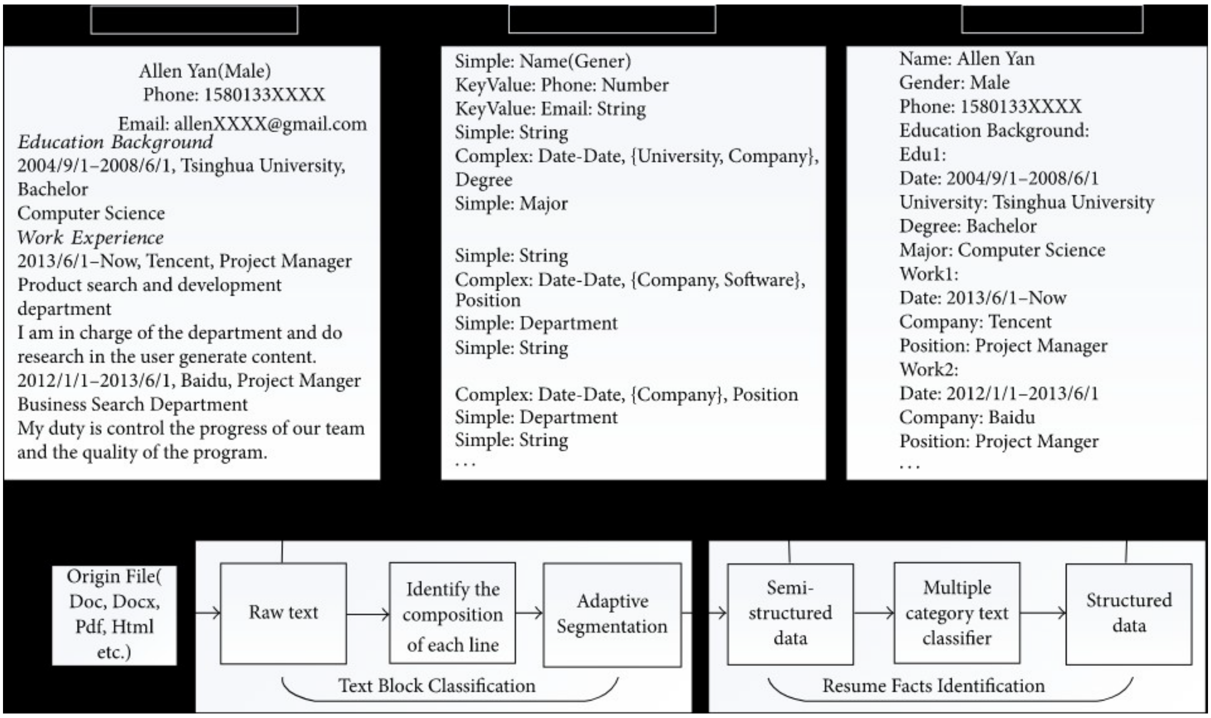
Country: United States

College: Fullstack Academy

Specialization: NLP

Problem description:

Resumes contain surfeit information irrelevant to the HR/authority, and they must manually process the resumes to shortlist the promising candidates. And thus, making the shortlisting task a herculean task for HR. Using the NER (Named Entity Recognition) model of NLP, this problem can be solved by finding and classifying the entities present in each resume into predefined classes such as person name, college name, academic information, relevant experiences, skill set, etc.



Business understanding:

Business objectives:

This project is dedicated to HR managers to help them:

- Converting hours of labor into seconds.
- Increase recruiters' efficiency and availability.
- Reducing the need for more employees.
- Avoiding errors.

Data science objectives:

- Identifying suitable technologies for our business objectives.

- Training and deploying fast and efficient Deep Learning models.

Project plan:

- The dataset (json format) has been provided by Data Glacier.
- One person (intern) will work on this project for one month.
- Code (Jupiter notebooks), PowerPoint presentation, and a final report will be delivered by the deadline of this project.
- The NER (Named Entity Recognition) model of NLP will be used to sort and classify resumes.

Key results:
An efficient NLP model used for resume sorting and classifying with high accuracy should be delivered by the end of this project.

(Model deployment through a Flask web app, if possible, before the deadline)

<u>Project lifecycle along with deadlines:</u>
Business understanding:19/07->21/07
Data exploring and understanding22/07->27/07
Data Cleansing and Transformation:28/07->03/08
Presentation and proposed modeling techniques:04/08->06/08
Model Selection and Model Building:07/08->13/08
Final présentation and report:14/08->15/08

<u>Data understanding:</u>
The data that has been provided is a json file.
This is what the data looks like:


```
{'content': 'Govardhana K\nSenior Software Engineer\n\nBengaluru, Karnataka, Karnataka - Email me on Indeed: indeed.com/r/Govardhana-K/\nb2
{'content': "Harini Komaravelli\nTest Analyst at Oracle, Hyderabad\n\nHyderabad, Telangana - Email me on Indeed: indeed.com/r/Harini-\nKoma
{'content': 'Hartej Kathuria\nData Analyst Intern - Oracle Retail\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Hartej-Kathuri
{'content': "Ijas Nizamuddin\nAssociate Consultant - State Street\n\nIrinchayam B.O, Kerala - Email me on Indeed: indeed.com/r/Ijas-\nNizam
{'content': 'Imgeeyaul Ansari\njava developer\n\nPune, Maharashtra - Email me on Indeed: indeed.com/r/Imgeeyaul-Ansari/a7be1cc43a434ac4\n\n
{'content': 'Jay Madhavi\nNavi Mumbai, Maharashtra - Email me on Indeed: indeed.com/r/Jay-\nMadhavi/1e7d0305af766bf6\n\nI look forward to b
{'content': "Jitendra Babu\nFI/CO Consultant in Tech Mahindra - SAP FICO\n\nChennai, Tamil Nadu - Email me on Indeed: indeed.com/r/Jitendra
{'content': 'Jyotirbindu Patnaik\nAssociate consultant@SAP labs , Bangalore Karnataka\n\nBengaluru, Karnataka - Email me on Indeed: indeed.
{'content': 'Karthihayini C\nSystems Engineer - Infosys Limited\n\nRajapalaiyam, Tamil Nadu - Email me on Indeed: indeed.com/r/Karthihayini
{'content': 'Karthik GV\nArchitect - Microsoft India\n\nHyderabad, Telangana - Email me on Indeed: indeed.com/r/Karthik-GV/1961c4eff806e6f4
{'content': "Kartik Sharma\nSystems Engineer - Infosys Ltd\n\nDelhi, Delhi - Email me on Indeed: indeed.com/r/Kartik-Sharma/cc7951fd7809f35
{'content': "Kasturika Borah\nTeam Member - Cisco\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Kasturika-\nBorah/9e71468914b3
{'content': "Kavitha K\nSenior System Engineer - Infosys Limited\n\nSalem, Tamil Nadu - Email me on Indeed: indeed.com/r/Kavitha-K/8977ce8c
{'content': 'Kavya U.\nNetwork Ops Associate - Accenture\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Kavya-U/049577580b3814e
{'content': 'Khushboo Choudhary\nDeveloper\n\nNoida, Uttar Pradesh - Email me on Indeed: indeed.com/r/Khushboo-Choudhary/\nb10649068fcdfa42
{'content': 'kimaya sonawane\nThane, Maharashtra - Email me on Indeed: indeed.com/r/kimaya-\nsonawane/1f27a18d2e4b1948\n\nQuality education
{'content': "Koushik Katta\nDevops\n\nHyderabad, Telangana - Email me on Indeed: indeed.com/r/Koushik-Katta/a6b19244854199ec\n\nDevOps Admi
{'content': 'Kowsick Somasundaram\nCertified Network Associate Training Program\n\nErode, Tamil Nadu - Email me on Indeed: indeed.com/r/Kow
{'content': 'Lakshika Neelakshi\nSenior Systems Engineer - Infosys Limited\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Laksh
{'content': "Madas Peddaiah\nAnantapur, Andhra Pradesh - Email me on Indeed: indeed.com/r/Madas-\nPeddaiah/557069069de72b14\n\n• Having 3 m
{'content': "Madhuri Sripathi\nBanglore, Karnataka, Karnataka - Email me on Indeed: indeed.com/r/Madhuri-\nSripathi/04a52a262175111c\n\nAro
{'content': "Mahesh Vijay\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Mahesh-Vijay/a2584aabc9572c30\n\nOver 6.5 years of funct
{'content': "Manisha Bharti\nSoftware Automation Engineer\n\nPune, Maharashtra - Email me on Indeed: indeed.com/r/Manisha-Bharti/3573e36088
{'content': "Manjari Singh\nSenior Software Analyst - Accenture\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Manjari-Singh/fd
{'content': 'Mohamed Ameen\nSystem engineer\n\nBengaluru, Karnataka - Email me on Indeed: indeed.com/r/Mohamed-Ameen/\nba052bfa70e4c0b7\n\n
```

It is composed of 200 resumes. Each line represents a two keys dictionary:

"annotation" is the key to the labeled resume, which looks like this:

```
print(data[0]['annotation'])

[{'label': ['Companies worked at'], 'points': [{'start': 1749, 'end': 1754, 'text': 'Oracle'}]}, {'label': ['Companies worked at'], 'points': [{'start': 1696, 'end': 1701, 'te
```

"content" is the key to the plain resume text, which looks like this:

```
print(data[0]["content"])

Govardhana K
Senior Software Engineer

Bengaluru, Karnataka, Karnataka - Email me on Indeed: indeed.com/r/Govardhana-K/
b2de315d95905b68

Total IT experience 5 Years 6 Months
Cloud Lending Solutions INC 4 Month • Salesforce Developer
Oracle 5 Years 2 Month • Core Java Developer
Languages Core Java, Go Lang
Oracle PL-SQL programming,
Sales Force Developer with APEX.

Designations & Promotions

Willing to relocate: Anywhere

WORK EXPERIENCE

Senior Software Engineer

Cloud Lending Solutions -  Bangalore, Karnataka -

January 2018 to Present

Present

Senior Consultant

Oracle -  Bangalore, Karnataka -
```

Each resume feature is represented with a dictionary:

```
dict_keys (['label', 'points'])
```

'points' is the key to a dictionary that looks like this:

```
[{'start': 1749, 'end': 1754, 'text': 'Oracle'}]
```

The data provided does not seem to have a few problems:
- There might be unnecessary spaces or punctuation in the text.
- Characters like "\n" and "\r" exist in the text.

I intend to fix those problems using simple Python commands and Spacy.

(Because it has Built-in visualizers for syntax and NER)

__Data cleaning:__

I will be first putting the json data into a more appropriate format, getting rid at the same time of the unnecessary spaces on the right and the left and only keeping the start point and end point of each label (I will not be needing the text itself).

This is what the annotated text looks like:

```
{'entities': [(1749, 1755, 'Companies worked at'),
  (1696, 1702, 'Companies worked at'),
  (1417, 1423, 'Companies worked at'),
  (1356, 1793, 'Skills'),
  (1209, 1215, 'Companies worked at'),
  (1136, 1247, 'Skills'),
  (928, 932, 'Graduation Year'),
  (858, 889, 'College Name'),
  (821, 856, 'Degree'),
  (787, 791, 'Graduation Year'),
  (744, 750, 'Companies worked at'),
  (722, 742, 'Designation'),
  (658, 664, 'Companies worked at'),
  (640, 656, 'Designation'),
  (574, 580, 'Companies worked at'),
  (555, 572, 'Designation'),
  (470, 493, 'Companies worked at'),
  (444, 468, 'Designation'),
  (308, 314, 'Companies worked at'),
  (234, 240, 'Companies worked at'),
  (175, 198, 'Companies worked at'),
  (93, 136, 'Email Address'),
  (39, 48, 'Location'),
  (13, 37, 'Designation'),
  (0, 12, 'Name')]}
```

I will now be replacing the "\n" with simple spaces:
This is how the plain text looks like now:

```
dt[0][0]

'Govardhana K Senior Software Engineer  Bengaluru, Karnataka, Karnataka - Email me on Indeed: indeed.com/r,
Salesforce Developer Oracle 5 Years 2 Month • Core Java Developer Languages Core Java, Go Lang Oracle PL-S
WORK EXPERIENCE  Senior Software Engineer  Cloud Lending Solutions -  Bangalore, Karnataka -  January 2018
Staff Consultant  Oracle -  Bangalore, Karnataka -  January 2014 to October 2016  Associate Consultant  Or
ing  Adithya Institute of Technology -  Tamil Nadu  September 2008 to June 2012  https://www.indeed.com/r/
```

Next, using regex and simple Python code, I will remove leading and trailing white spaces from entity spans. This is the final output:

```
data[0][0]
```

```
'Govardhana K Senior Software Engineer  Bengaluru, Karnataka, Karnataka - Email m
Salesforce Developer Oracle 5 Years 2 Month • Core Java Developer Languages Core
WORK EXPERIENCE  Senior Software Engineer  Cloud Lending Solutions -  Bangalore,
Staff Consultant  Oracle -  Bangalore, Karnataka -  January 2014 to October 2016
ing  Adithya Institute of Technology -  Tamil Nadu  September 2008 to June 2012
K/b2de315d95905b68?isid=rex-download&ikw=download-top&co=IN   SKILLS  APEX. (Less
m/in/govardhana-k-61024944/  ADDITIONAL INFORMATION  Technical Proficiency:  Lang
r, NetBeans, Eclipse, SQL developer, PL/SQL Developer, WinSCP, Putty Web Technolc
ddleware: Web logic, OC4J Product FLEXCUBE: Oracle FLEXCUBE Versions 10.x, 11.x a
```

```
data[0][1]
```

```
{'entities': [[1749, 1755, 'Companies worked at'],
  [1696, 1702, 'Companies worked at'],
  [1417, 1423, 'Companies worked at'],
  [1356, 1793, 'Skills'],
  [1209, 1215, 'Companies worked at'],
  [1136, 1247, 'Skills'],
  [928, 932, 'Graduation Year'],
  [858, 889, 'College Name'],
  [821, 856, 'Degree'],
  [787, 791, 'Graduation Year'],
  [744, 750, 'Companies worked at'],
  [722, 742, 'Designation'],
  [658, 664, 'Companies worked at'],
  [640, 656, 'Designation'],
  [574, 580, 'Companies worked at'],
  [555, 572, 'Designation'],
  [470, 493, 'Companies worked at'],
  [444, 468, 'Designation'],
  [308, 314, 'Companies worked at'],
  [234, 240, 'Companies worked at'],
  [175, 198, 'Companies worked at'],
  [93, 136, 'Email Address'],
  [39, 48, 'Location'],
```

The entities of the data are now in lists, which makes them easier to iterate and use.

EDA:

To provide meaningful insights by analyzing the resume extraction dataset, I created a data frame containing the different labels of each unordered resume.

0	Companies worked at	oracle
1	Companies worked at	oracle
2	Companies worked at	oracle
3	Skills	languages: core java, go lang, data structures...
4	Companies worked at	oracle
...	...	...
3203	Degree	b- tech
3204	Designation	security analyst
3205	Companies worked at	infosys - career contour
3206	Designation	security analyst
3207	Name	pradeep kumar

3208 rows × 2 columns

I will be performing statistical analysis on each one of these elements except for the name, email, and designation:

```
print(df[0].unique())
```

```
['Companies worked at' 'Skills' 'Graduation Year' 'College Name' 'Degree'
 'Designation' 'Email Address' 'Location' 'Name' 'Years of Experience'
```

So, I split the data according to each label:

	0	text
6	Graduation Year	2012
9	Graduation Year	2012
56	Graduation Year	2016
59	Graduation Year	2018
70	Graduation Year	2009
...	...	...
3127	Graduation Year	2005
3130	Graduation Year	2013
3135	Graduation Year	2013
3136	Graduation Year	2013
3170	Graduation Year	2002

222 rows × 2 columns

0	Companies worked at	oracle
1	Companies worked at	oracle
2	Companies worked at	oracle
4	Companies worked at	oracle
10	Companies worked at	oracle
...	...	...
3186	Companies worked at	infosys bpo ltd
3189	Companies worked at	infosys bpo ltd
3192	Companies worked at	infosys bpo ltd
3194	Companies worked at	infosys bpo ltd
3205	Companies worked at	infosys - career contour

676 rows × 2 columns

	0	text
22	Location	bengaluru
35	Location	hyderabad
38	Location	hyderabad
39	Location	hyderabad
40	Location	hyderabad
...	...	...
3183	Location	bengaluru
3185	Location	bengaluru
3188	Location	bengaluru
3191	Location	bengaluru
3197	Location	bengaluru

381 rows × 2 columns

	0	text
3	Skills	languages: core java, go lang, data structures...
5	Skills	apex. (less than 1 year), data structures (3 y...
28	Skills	functional testing, blue prism, qtp
53	Skills	languages & technologies: python, r, sql, nos...
55	Skills	python (2 years), sql. (1 year), nosql (1 year...
...	...	...
3124	Skills	sap hana (4 years), sap ui5/fiori (4 years), a...
3155	Skills	data backup (1 year), exchange (1 year), lan (...
3163	Skills	auditing (less than 1 year), cfa (less than 1 ...
3169	Skills	excel (10+ years), operations (7 years), proje...
3201	Skills	splunk, network security, arc sight (2 years),...

417 rows × 2 columns

I also prepared my data frame to make it easier to use with the matplotlib and the seaborn libraries.

Transforming all the text into lowercase, removing unnecessary spaces, changing dates into numeric variables, and removing unnecessary words.

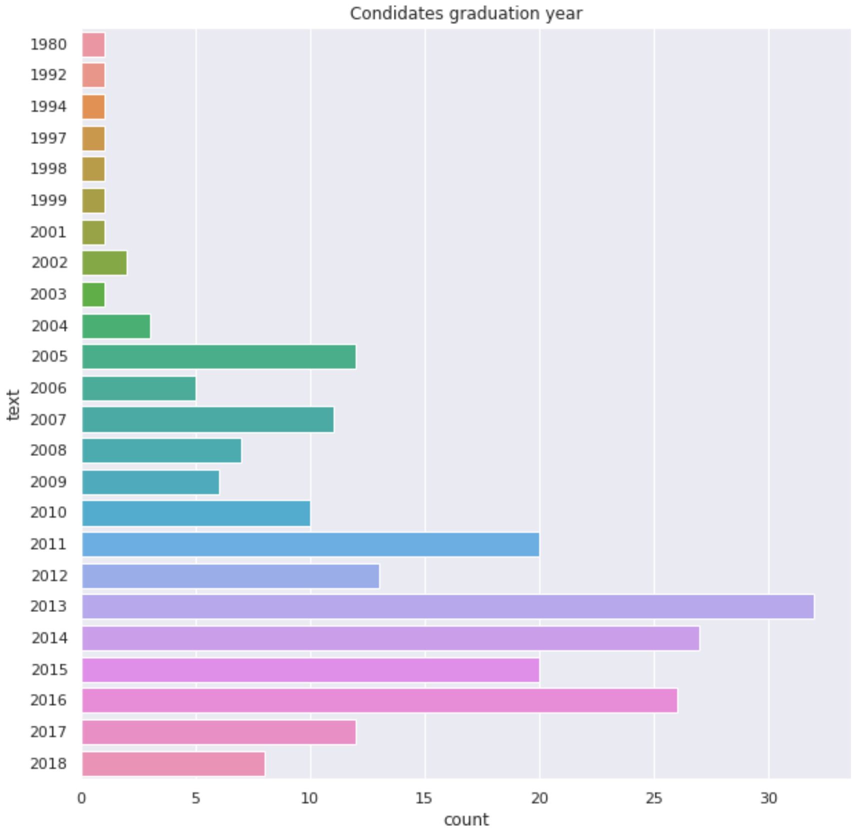
```
0                           oracle
1                           oracle
2                           oracle
4                           oracle
10                          oracle
                         ...
3186              infosys bpo ltd
3189              infosys bpo ltd
3192              infosys bpo ltd
3194              infosys bpo ltd
3205     infosys - career contour
Name: text, Length: 676, dtype: object
```

```
[ ] stopwords = ['what', 'who', 'is', 'a', 'at', 'is', 'he','of','university','college','public','private','school','institute','academy']
    L4=CollegeN["text"]
    L=list(L4)
    H=[]
    L3=[]
    for i in range(291):
      H=L[i].split()
      L1 = [word for word in H if word.lower() not in stopwords]
      L2= ' '.join(L1)
      L3.append(L2)

    print(L3)

    ['adithya', 'osmania', 'osmania', 'manipal', 'manipal', '', 'birla', 'rashtriya military bangalore', 'rashtriya military bangalore', 'army', 'acharya chembur',
```

```
Gradyear["text"]=pd. to_numeric(Gradyear["text"])
type(Gradyear['text'][6])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py
A value is trying to be set on a copy of a slice from a Data
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.
  """Entry point for launching an IPython kernel.
numpy.int64
```

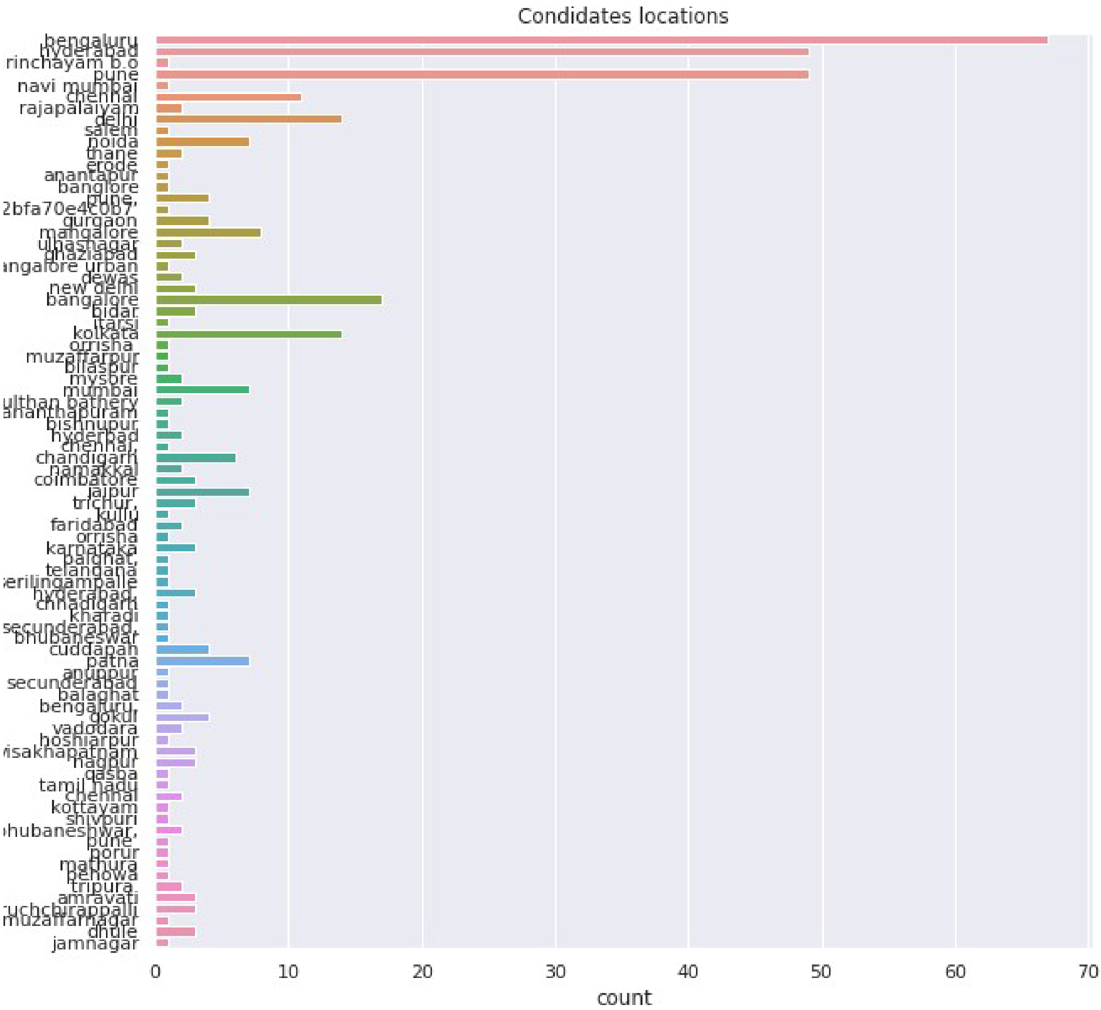
Starting with the graduation year:



We can see that most of the candidates graduated after 2005.

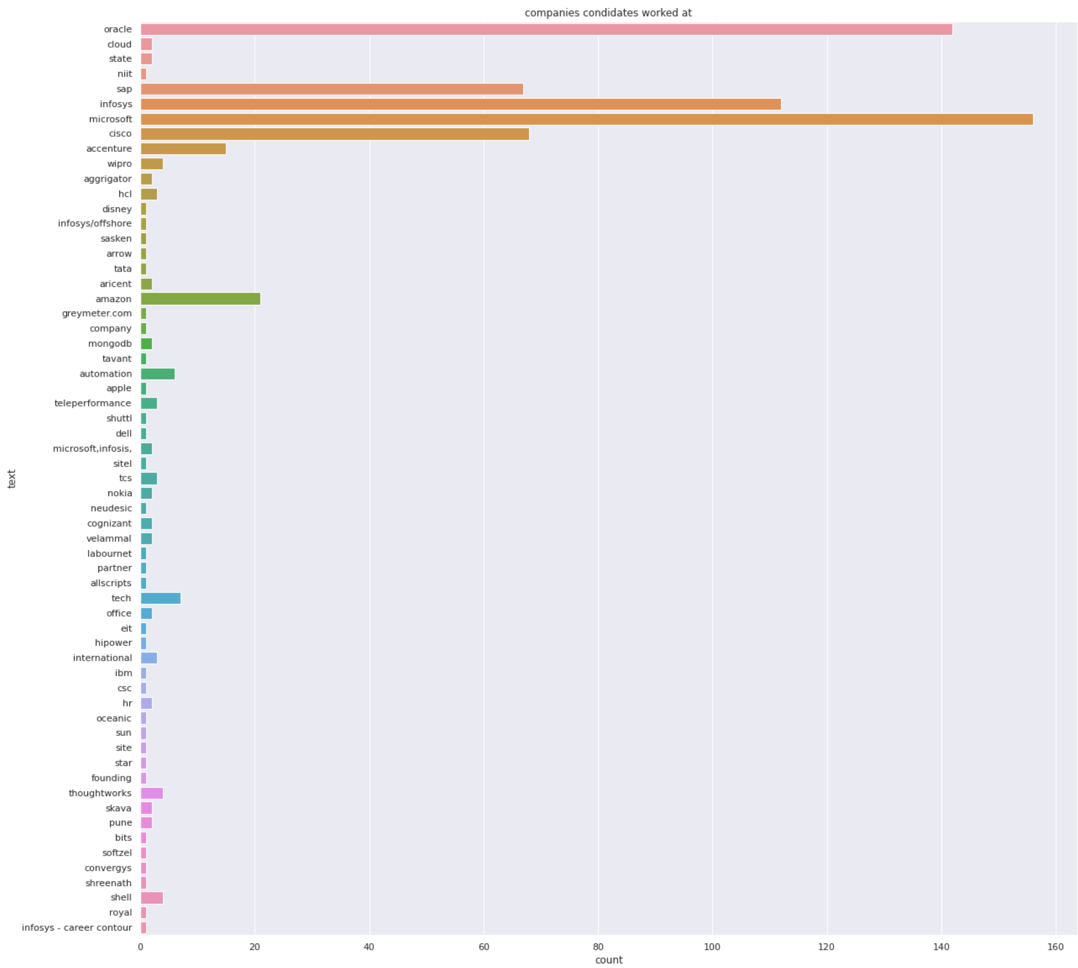
16% of the candidates graduated in 2013 only, and 50% graduated between 2016 and 2013. No candidates graduated after 2018.

Next, I have candidates' locations:

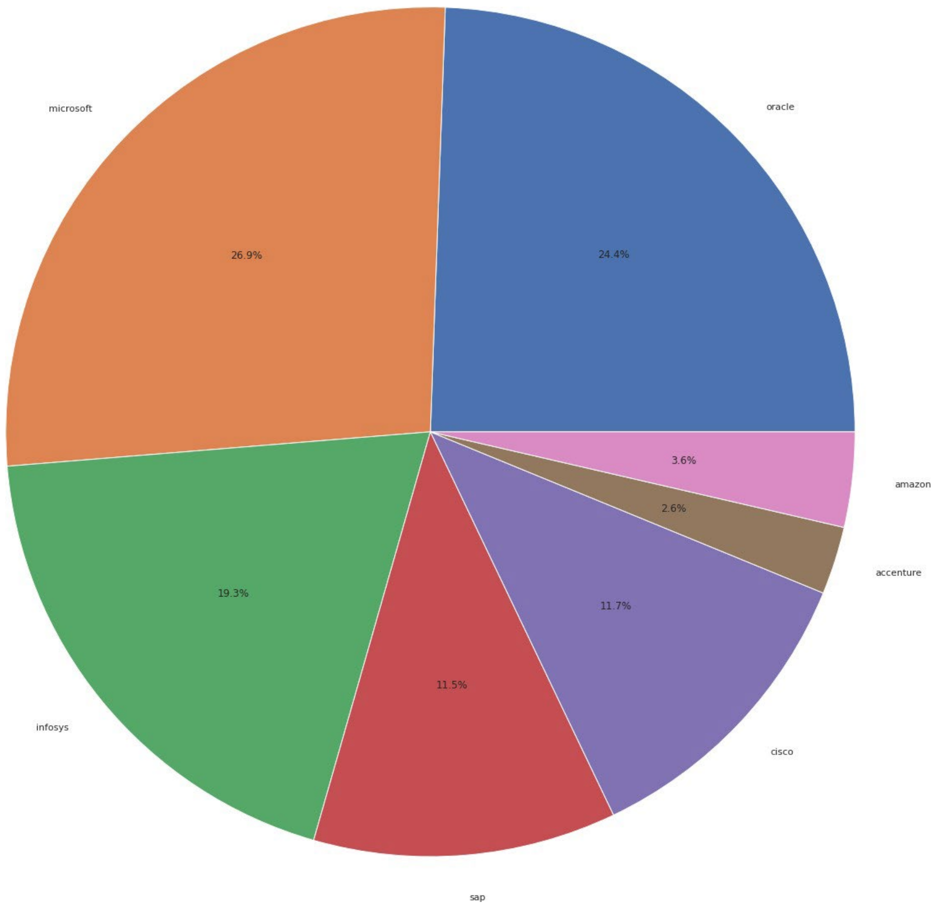


Most candidates are in Bengaluru (67 out of 200), Hyderabad and Pune (49 in each).

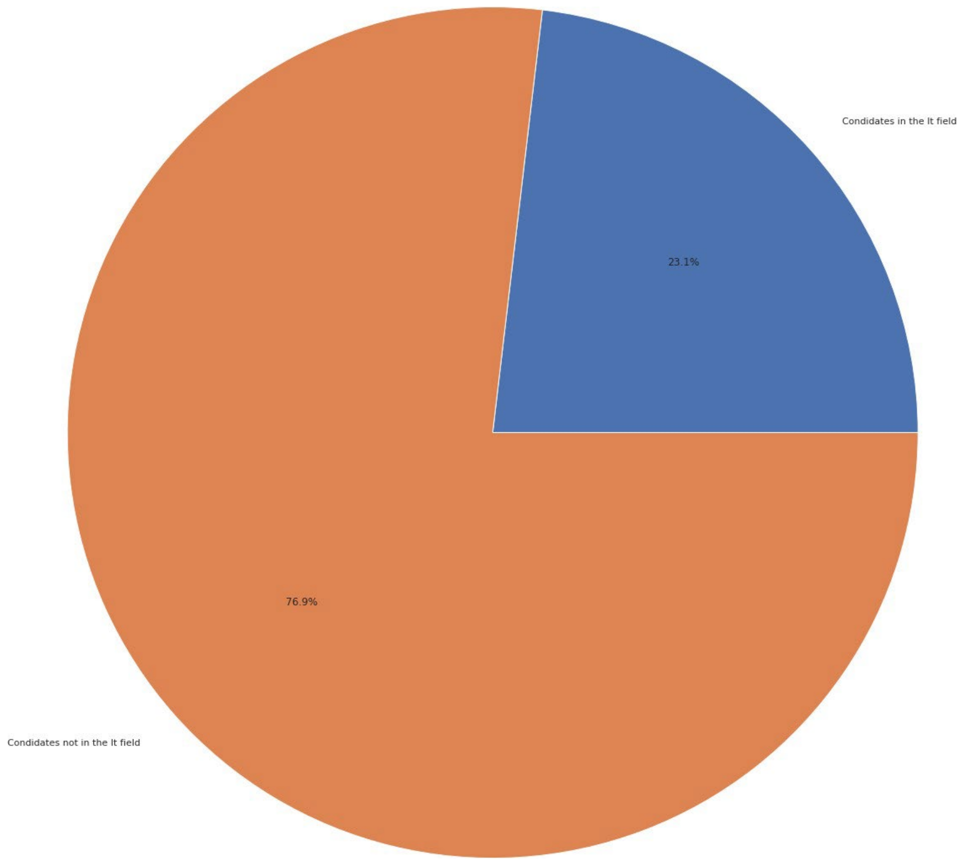
As for the 'Companies worked at' data frame, the following chart shows that Oracle, Microsoft, SAP, Cisco, NIIT, and Infosys are the ones with the biggest numbers of candidates having worked at them in the past.



At least 155 out of 200 (26.9%, as seen through this chart below) candidates worked at Microsoft before, and 142 out of 200(24.4%) worked at Oracle.



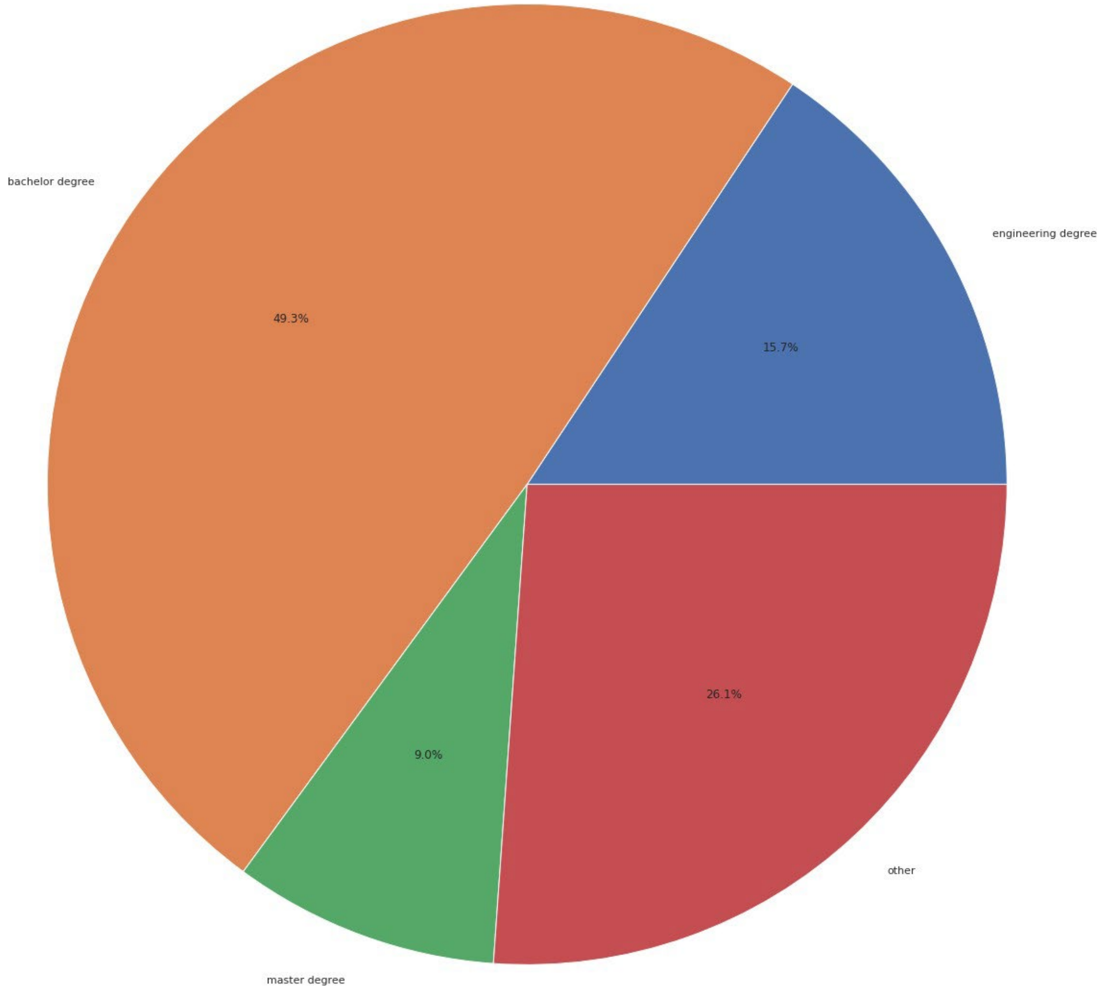
Analyzing the "Degree" data frame showed me that only 23.1% of the candidates for this job are in IT, while the rest have different fields of study, such as business chemistry, electronics, etc.



49.3% of the candidates have a bachelor's degree.

15.7% of them are engineers.

9% of candidates have a master's degree.



Analyzing the universities, the candidates studied at, I figured out that almost everyone went to a different college.

```
CollegeN["text"]=L3
len(CollegeN["text"].unique())
```

```
/usr/local/lib/python3.7/dist-packages
A value is trying to be set on a copy
Try using .loc[row_indexer,col_indexer

See the caveats in the documentation:
  """Entry point for launching an IPyt
238
```

Depending on these insights, the client's HR department can request the elimination of some candidates' categories depending on the job profile needed (for example, the client needs candidates with engineering degrees)

Also, this EDA has shown that those who applied for this job are quite different, especially when talking about the fields of study; I recommend that the HR department takes more care of the job description and the requirements provided to have more accurate candidates' resumes.

Model building:

I use the BERT model to build the resume extraction depending on our data, business, and data science objectives.

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique developed by Google for natural language processing (NLP) pre-training.

I started with changing data to the appropriate format to feed it to the model:

```
[ ] # Changing data to appropriate format so as to feed it to the model
    from tqdm import tqdm_notebook as tqdm
    import pandas as pd
    from tqdm import tqdm,trange
    cleanedDF = pd.DataFrame(columns=["setences_cleaned"])
    sum1 = 0
    for i in tqdm(range(len(dt))):
        start = 0
        emptyList = ["Empty"] * len(dt[i][0].split())
        numberOfWords = 0
        lenOfString = len(dt[i][0])
        strData = dt[i][0]
        strDictData = dt[i][1]
        lastIndexOfSpace = strData.rfind(' ')
        for i in range(lenOfString):
            if (strData[i]==" " and strData[i+1]!=" "):
                for k,v in strDictData.items():
                    for j in range(len(v)):
                        entList = v[len(v)-j-1]
                        if (start>=int(entList[0]) and i<=int(entList[1])):
                            emptyList[numberOfWords] = entList[2]
                            break
                        else:
                            continue
                start = i + 1
                numberOfWords += 1
            if (i == lastIndexOfSpace):
                for j in range(len(v)):
                        entList = v[len(v)-j-1]
                        if (lastIndexOfSpace>=int(entList[0]) and lenOfString<=int(entList[1])):
                            emptyList[numberOfWords] = entList[2]
                            numberOfWords += 1
        cleanedDF = cleanedDF.append(pd.Series([emptyList],  index=cleanedDF.columns ), ignore_index=True )
        sum1 = sum1 + numberOfWords

    100%|██████████| 200/200 [00:01<00:00, 118.64it/s]
```

The result looks like this:

```
[ ] cleanedDF.head()
```

	setences_cleaned
0	[Name, Name, Designation, Designation, Empty, ...
1	[Name, Name, Designation, Designation, Empty, ...
2	[Name, Name, Designation, Designation, Designa...
3	[Name, Name, Designation, Designation, Empty, ...
4	[Name, Name, Designation, Empty, Empty, Empty,...

Next, text tokenization:

```
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=20000)
tokenizer.fit_on_texts(data["content"])
```

```
[ ] for i in range(len(data)):
      tokenized_texts = tokenizer.texts_to_sequences(data["content"])

    print(tokenized_texts[0])

    [1979, 893, 187, 49, 60, 168, 73, 73, 62, 57, 8, 11, 11, 13, 16, 1979, 893, 3170, 569,
```

Initializing the model's parameters:

```
input_ids = pad_sequences(tokenized_texts,
                          maxlen=MAX_LEN, dtype="long", truncating="post", padding="post")
```

```
[ ] MAX_LEN = 300
    bs = 16
```

```
[ ] device = torch.device("cuda")
    n_gpu = torch.cuda.device_count()
```

```
[ ] tags_vals = ["UNKNOWN", "Name", "Degree","Skills","College Name","Email Address","Designation","Companies worked at",
    tag2idx = {t: i for i, t in enumerate(tags_vals)}
```

```
[ ] labels = cleanedDF["setences_cleaned"].tolist()
```

```
[ ] tags = pad_sequences([[tag2idx.get(l) for l in lab] for lab in labels],
                         maxlen=MAX_LEN, value=tag2idx["Empty"], padding="post",
                         dtype="long", truncating="post")
```

```
[ ] attention_masks = [[float(i>0) for i in ii] for ii in input_ids]
```

Splitting data to test and train sets:

```
[ ] from sklearn.model_selection import train_test_split
    tr_inputs, val_inputs, tr_tags, val_tags = train_test_split(input_ids, tags,
                                                                random_state=2018, test_size=0.1)
    tr_masks, val_masks, _, _ = train_test_split(attention_masks, input_ids,
                                                 random_state=2018, test_size=0.1)
```

Transforming the data to be fed to the model to torch tensors:

```
[ ] tr_inputs = torch.tensor(tr_inputs)
    val_inputs = torch.tensor(val_inputs)
    tr_tags = torch.tensor(tr_tags)
    val_tags = torch.tensor(val_tags)
    tr_masks = torch.tensor(tr_masks)
    val_masks = torch.tensor(val_masks)
```

Loading data and building the model:

```
[ ] train_data = TensorDataset(tr_inputs, tr_masks, tr_tags)
    train_sampler = RandomSampler(train_data)
    train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=bs)

    valid_data = TensorDataset(val_inputs, val_masks, val_tags)
    valid_sampler = SequentialSampler(valid_data)
    valid_dataloader = DataLoader(valid_data, sampler=valid_sampler, batch_size=bs)
```

```
[ ] model.cuda();
```

```
model = BertForTokenClassification.from_pretrained("bert-base-uncased", num_labels=len(tag2idx))
```

```
[ ] FULL_FINETUNING = True
    if FULL_FINETUNING:
        param_optimizer = list(model.named_parameters())
        no_decay = ['bias', 'gamma', 'beta']
        optimizer_grouped_parameters = [
            {'params': [p for n, p in param_optimizer if not any(nd in n for nd in no_decay)],
             'weight_decay_rate': 0.01},
            {'params': [p for n, p in param_optimizer if any(nd in n for nd in no_decay)],
             'weight_decay_rate': 0.0}
        ]
    else:
        param_optimizer = list(model.classifier.named_parameters())
        optimizer_grouped_parameters = [{"params": [p for n, p in param_optimizer]}]
    optimizer = Adam(optimizer_grouped_parameters, lr=3e-5)
```

Model training, with 5 epochs:

```
[ ] epochs = 5
    max_grad_norm = 1.0
    import numpy as np
    for _ in trange(epochs, desc="Epoch"):
        # TRAIN loop
        model.train()
        tr_loss = 0
        nb_tr_examples, nb_tr_steps = 0, 0
        for step, batch in enumerate(train_dataloader):
            # add batch to gpu
            batch = tuple(t.to(device) for t in batch)
            b_input_ids, b_input_mask, b_labels = batch
            # forward pass
            loss = model(b_input_ids, token_type_ids=None,
                         attention_mask=b_input_mask, labels=b_labels)
            # backward pass
            loss.backward()
            # track train loss
            tr_loss += loss.item()
            nb_tr_examples += b_input_ids.size(0)
            nb_tr_steps += 1
            # gradient clipping
            torch.nn.utils.clip_grad_norm_(parameters=model.parameters(), max_norm=max_grad_norm)
            # update parameters
            optimizer.step()
            model.zero_grad()
        # print train loss per epoch
        print("Train loss: {}".format(tr_loss/nb_tr_steps))
        # VALIDATION on validation set
        model.eval()
        eval_loss, eval_accuracy = 0, 0
        nb_eval_steps, nb_eval_examples = 0, 0
        predictions , true_labels = [], []
        for batch in valid_dataloader:
            batch = tuple(t.to(device) for t in batch)
            b_input_ids, b_input_mask, b_labels = batch
```

This is the output of the training:

```
Epoch:   0%|          | 0/5 [00:00<?, ?it/s]Train loss: 0.8996409103274345
Epoch:  20%|██        | 1/5 [00:11<00:46, 11.65s/it]Validation loss: 0.5537128895521164
Validation Accuracy: 0.9138541666666666
F1-Score: 0.9417320178767181
Train loss: 0.5108052641153336
Epoch:  40%|████      | 2/5 [00:23<00:35, 11.67s/it]Validation loss: 0.517583355307579
Validation Accuracy: 0.9138541666666666
F1-Score: 0.9417320178767181
Train loss: 0.4752659574151039
Epoch:  60%|██████    | 3/5 [00:35<00:23, 11.75s/it]Validation loss: 0.4547186344861984
Validation Accuracy: 0.9138541666666666
F1-Score: 0.9417320178767181
Train loss: 0.4253872831662496
Epoch:  80%|████████  | 4/5 [00:47<00:11, 11.83s/it]Validation loss: 0.42342111468315125
Validation Accuracy: 0.9171875
F1-Score: 0.9430812041487477
Train loss: 0.4088966970642407б
Epoch: 100%|██████████| 5/5 [00:59<00:00, 11.90s/it]Validation loss: 0.4165296256542206
Validation Accuracy: 0.9138541666666666
F1-Score: 0.9417320178767181
```

As we can see, the BERT model has 0.91 accuracy.

<u>Dealing with resumes in other formats:</u>

In this part, I will be using two packages: pdfminer. six, doc2text, and python-magic.

So, I am starting with installing them following:

```
pip install pdfminer.six
```

```
pip install python-magic
```

```
pip install doc2text
```

Then, I will create a function that mines text from pdf files (extract_text_from_pdf).

```
import io
def extract_text_from_pdf(pdf_path):
    with open(pdf_path, 'rb') as fh:
        # iterate over all pages of PDF document
        for page in PDFPage.get_pages(fh, caching=True, check_extractable=True):
            # creating a resoure manager
            resource_manager = PDFResourceManager()

            # create a file handle
            fake_file_handle = io.StringIO()

            # creating a text converter object
            converter = TextConverter(
                                resource_manager,
                                fake_file_handle,
                                codec='utf-8',
                                laparams=LAParams()
                        )

            # creating a page interpreter
            page_interpreter = PDFPageInterpreter(
                                resource_manager,
                                converter
                            )

            # process current page
            page_interpreter.process_page(page)

            # extract text
            text = fake_file_handle.getvalue()
            yield text

            # close open handles
            converter.close()
            fake_file_handle.close()
```

I did the same for doc files:

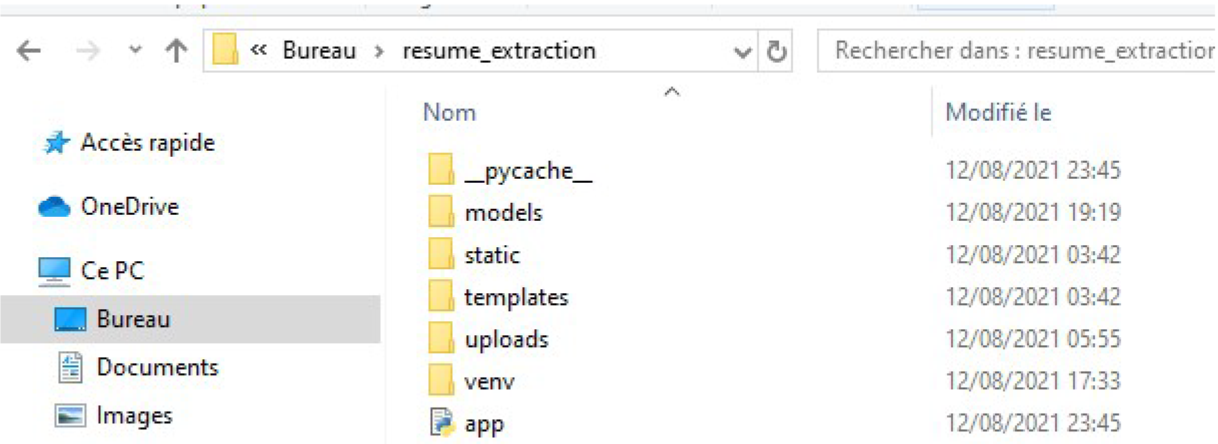
```
import docx2txt

def extract_text_from_doc(doc_path):
    temp = docx2txt.process(doc_path)
    return temp
```

Saving the model:

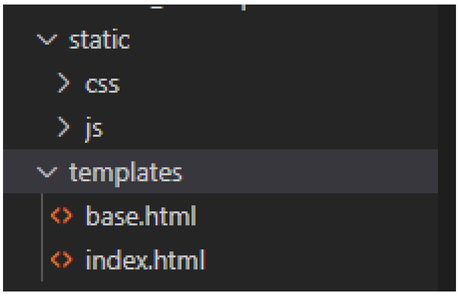
```
torch.save(model.state_dict(),"saved_model.pb")
```

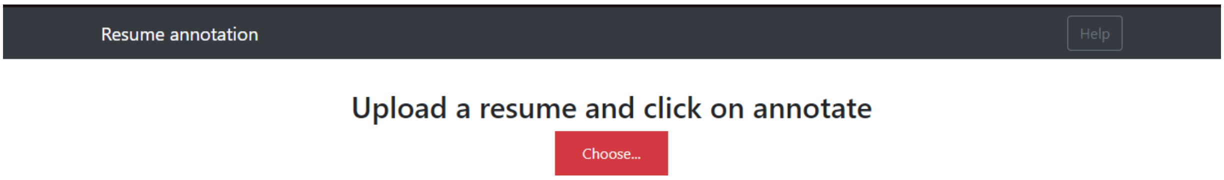
<u>Deployment:</u>

I started with creating a Flask web app:



I put together the front-end part using only HTML, CSS, and JS:





Importing necessary modules:

```
from keras.preprocessing.sequence import pad_sequences
from seqeval.metrics import classification_report,accuracy_score,f1_score
import math
import torch
import os
from torch.optim import Adam
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

from pytorch_pretrained_bert import BertTokenizer, BertConfig
from pytorch_pretrained_bert import BertForTokenClassification, BertAdam
import pandas as pd
import numpy as np
import re
from tqdm import tqdm_notebook as tqdm
from tqdm import tqdm,trange
import docx2txt
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
import io
import json
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
import pickle
```

Defining the app:

```
# Define a flask app
app = Flask(__name__)
```

Loading the model:

```
PATH = (r'models\saved_model.pb')
# Load your trained model
model = BertForTokenClassification.from_pretrained("bert-base-uncased",num_labels=12)

model.load_state_dict(torch.load(PATH, map_location='cpu'))
model.eval()

print('Model loaded. Check http://127.0.0.1:5000/')
```

Creating a function for text mining from files depending on the format:

```
def prepare_data(path):
    name, extension = os.path.splitext(path)
    text=""
    if extension==".pdf":
        for page in extract_text_from_pdf(path):
            text += ' ' + page
            return text
    elif extension==".docx":
        extract_text_from_doc(path)
        return text
    elif extension==".json":
        text=json.load(path)
        return text
    else:
        return "File not supported"
```

Creating a function that feeds the model the text data to predict an annotation for it:

```
def predict(text):

    text=text.replace("\n"," ")
    tokenizer = Tokenizer(num_words=20000)
    tokenizer.fit_on_texts(text)
    tokenized_texts = tokenizer.texts_to_sequences(text)
    input_ids = pad_sequences(tokenized_texts,maxlen=10000, dtype="long", truncating="post", padding="post")
    text1= torch.tensor(input_ids)
    device = torch.device("cuda")
    logits = model(text1,token_type_ids=None)
    logits = logits.detach().cpu().numpy()
    predictions=[]
    tags_vals = ["UNKNOWN", "Name", "Degree","Skills","College Name","Email Address","Designation","Companies worked
    predictions.extend([list(p) for p in np.argmax(logits, axis=2)])
    pred_tags = [tags_vals[p_i] for p in predictions for p_i in p]
    result=""
    for i in range(len(pred_tags)):
        result+=str(pred_tags[i])
    return str(result)
```

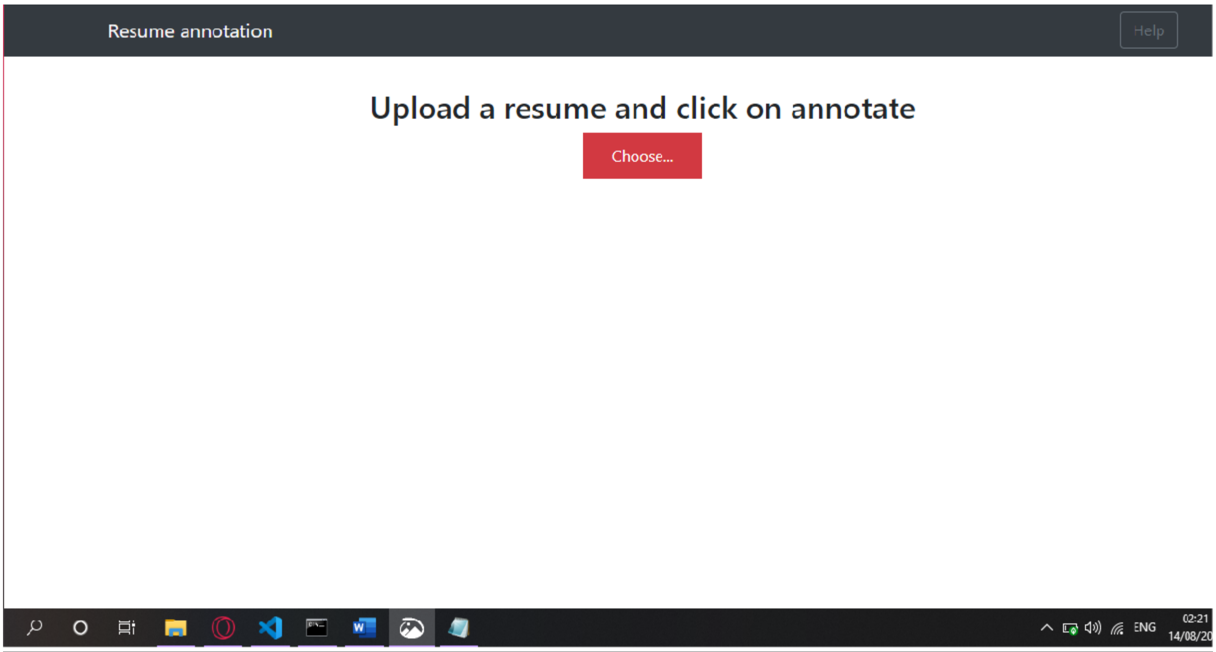
The primary page function:

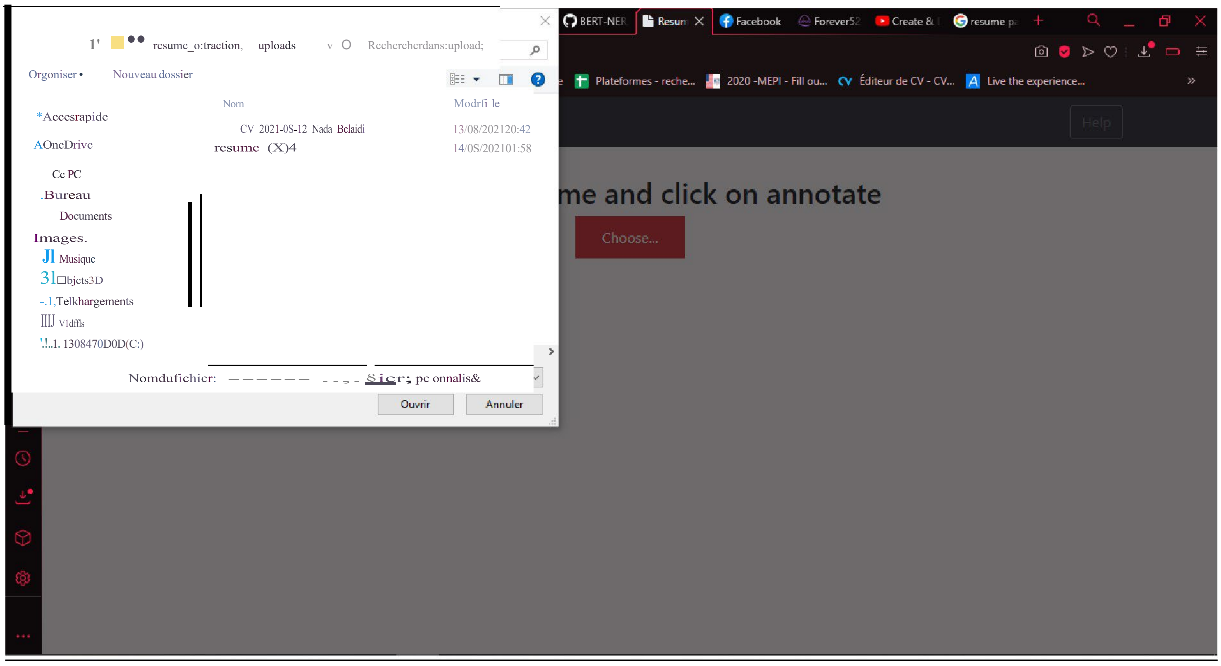
```
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
```

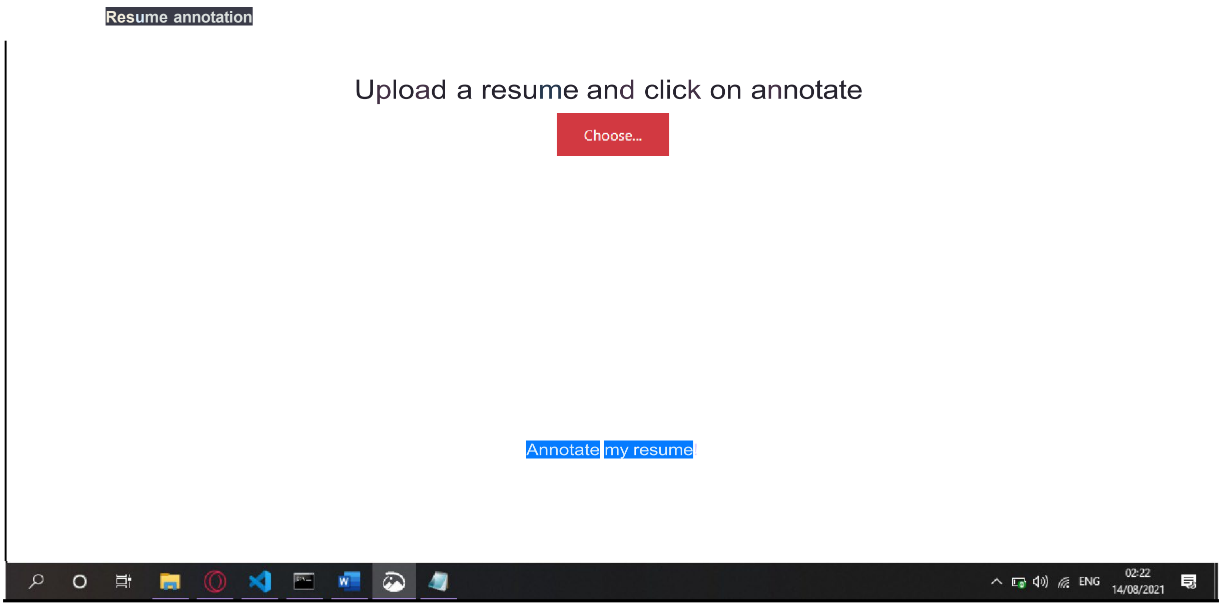
A function that uploads and saves files and uses them for predictions:

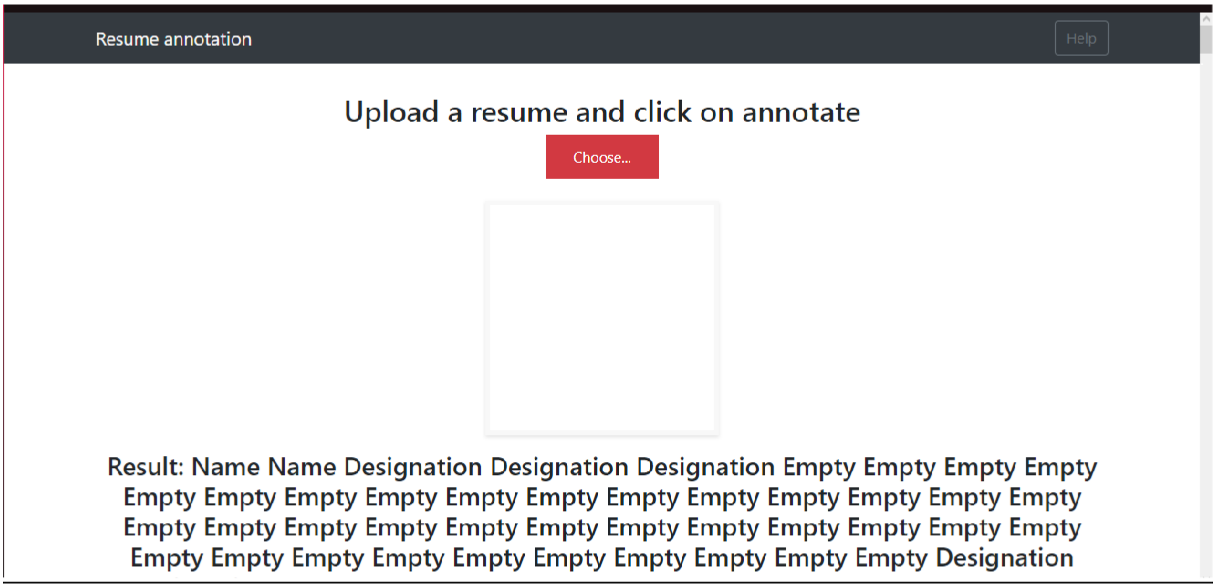
```python
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        text = prepare_data(file_path)
        result=predict(text)
        return result
    return None
```

Demo:




resume_extraction › uploads
Organiser ▾
Nouveau dossier
Nom
Modifié le
Accès rapide
OneDrive
Ce PC
Bureau
Documents
Images
Musique
Objets 3D
Téléchargements
Vidéos
CV_2021-08-12_Nada_Belaidi
13/08/2021 20:42
resume_(X)4
14/08/2021 01:58
Nom du fichier:
Ouvrir
Annuler
BERT-NER
Facebook
Create &
resume p
Plateformes - reche...
2020 -MEPI - Fill ou...
Éditeur de CV - CV...
Live the experience...
Help
me and click on annotate
Choose...


Resume annotation
Upload a resume and click on annotate
Choose...
Annotate my resume
ENG
02:22
14/08/2021



__GitHub Repo link:__ https://github.com/NadaBelaidi/NLP-Resume-Extraction