Data Glacier

Your Deep Learning Partner

NLP: Resume Extraction

Virtual Internship

Agenda

Team member’s details

Problem description

Approach

EDA

Recommendations

Model training

Results

Team member’s details

- Group Name: NLP: Resume Extraction

- Name: Tatiana Moteu Ngoli
- Email: mtatiana@aimsammi.org
- Country: Germany

- Github repo link: https://github.com/TatianaMoteuN/Data-Glacier/tree/master/week13

Problem description

Resumes contain surfeit information that is not relevant for the HR/authority, and they have to manually process the resumes to shortlist the promising candidates for them. And, thus making the shortlisting task a herculean task for HR. By making use of the NER(Named Entity Recognition) model of NLP this problem can be solved by finding and classifying the entities that are present in each resume into predefined classes such as person name, college name, academics information, relevant experiences, skill set, etc.

Task

- Problem Understanding
- Data annotation
- Named Entity Recognition (NER)
- Model building & training
- Performance evaluation & reporting
- Model Deployment
- Model Inference

Data Exploration

- 2 columns: content & annotation
- Total data points :160
- 18 NER tags found: ORDINAL, WORK_OF_ART, NORP, GPE, FAC, TIME, ORG, DATE,LANGUAGE, PRODUCT,
- PERCENT, MONEY, LAW, EVENT, PERSON, QUANTITY

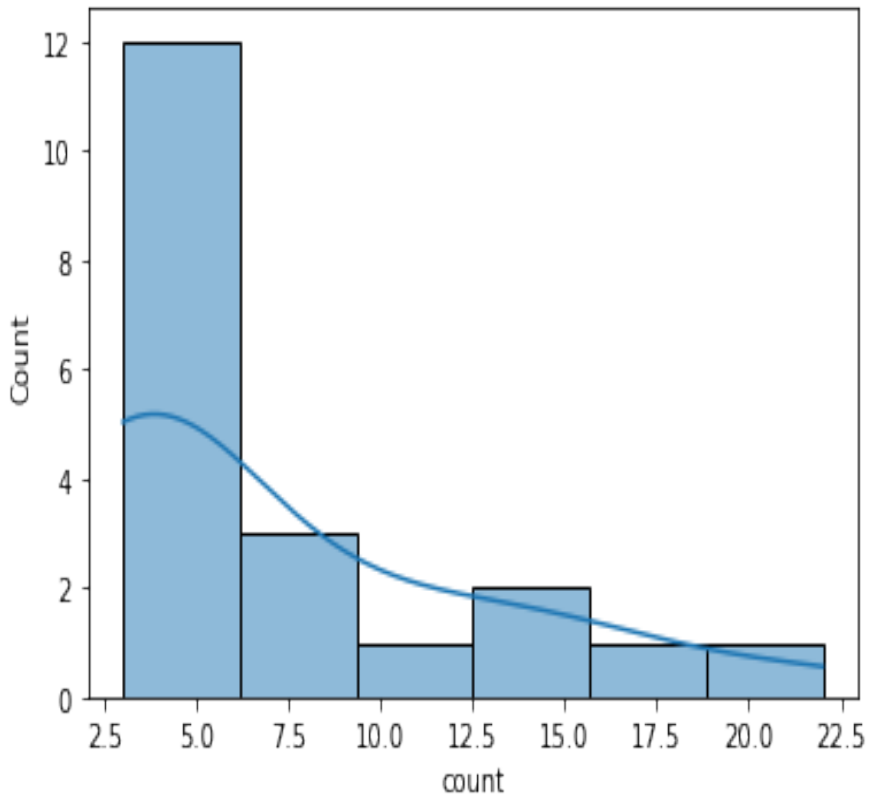
Assumptions:

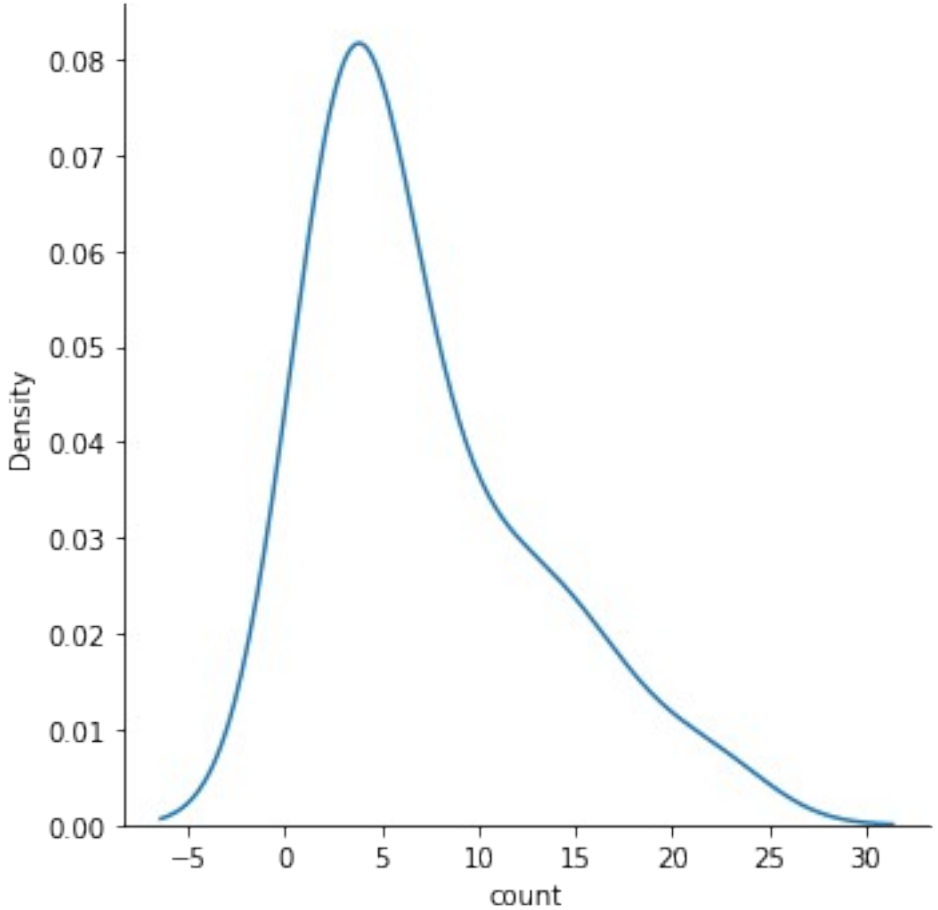
- We need to know what kind of person have applied in that company.

- The language speak by those persons

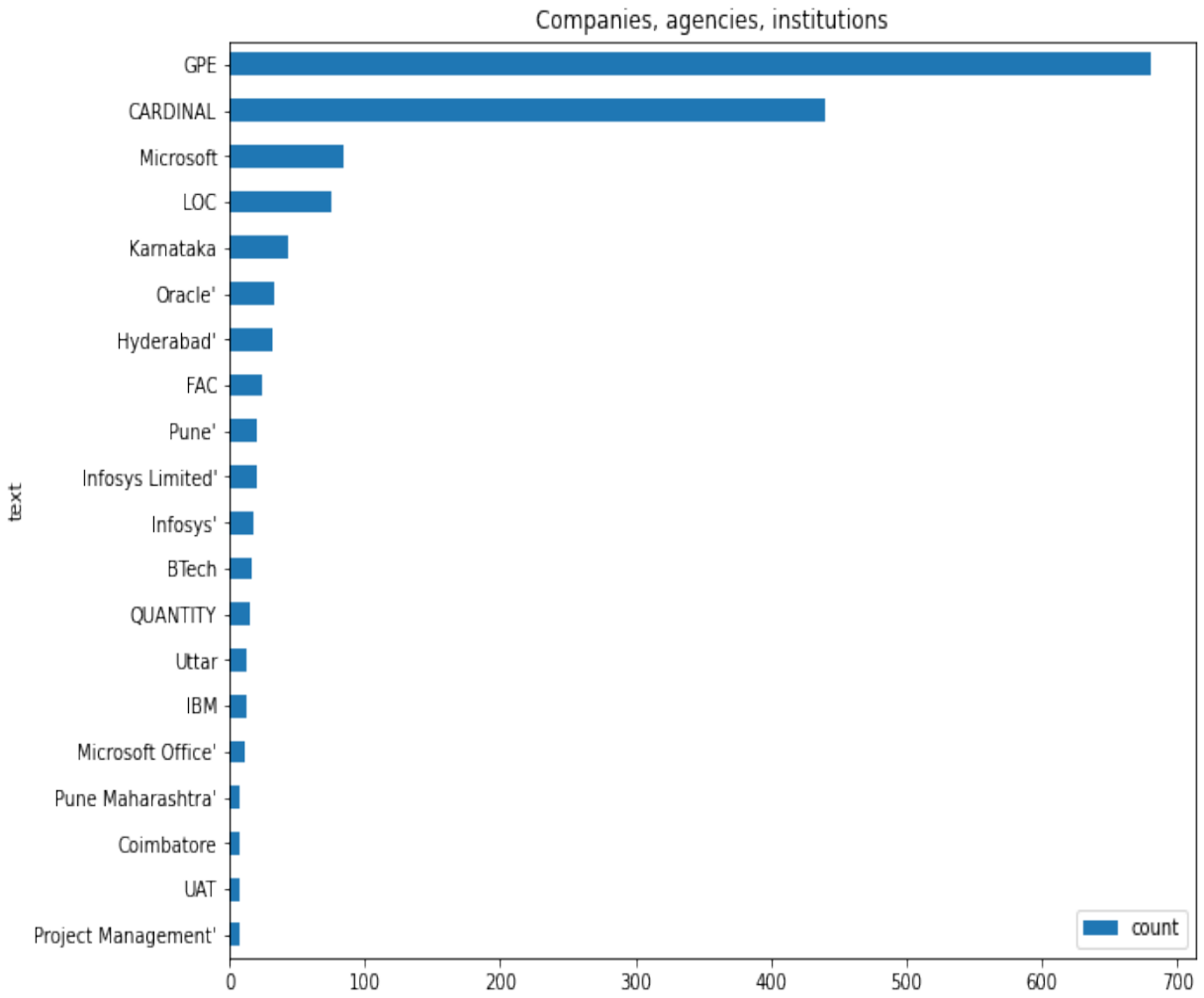
- Their past works
- And where they come from

Density PERSON Analysis



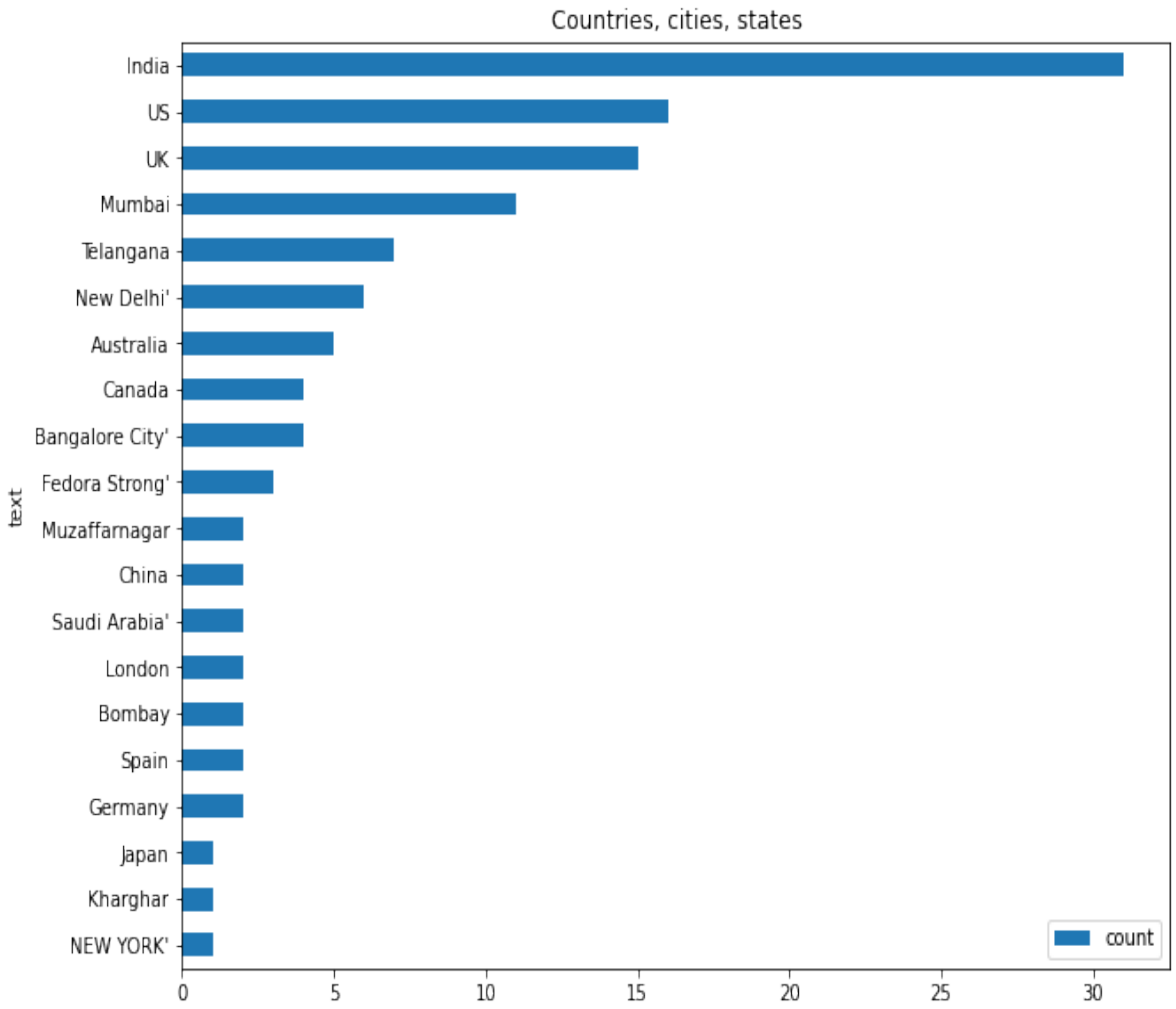


ORG Analysis



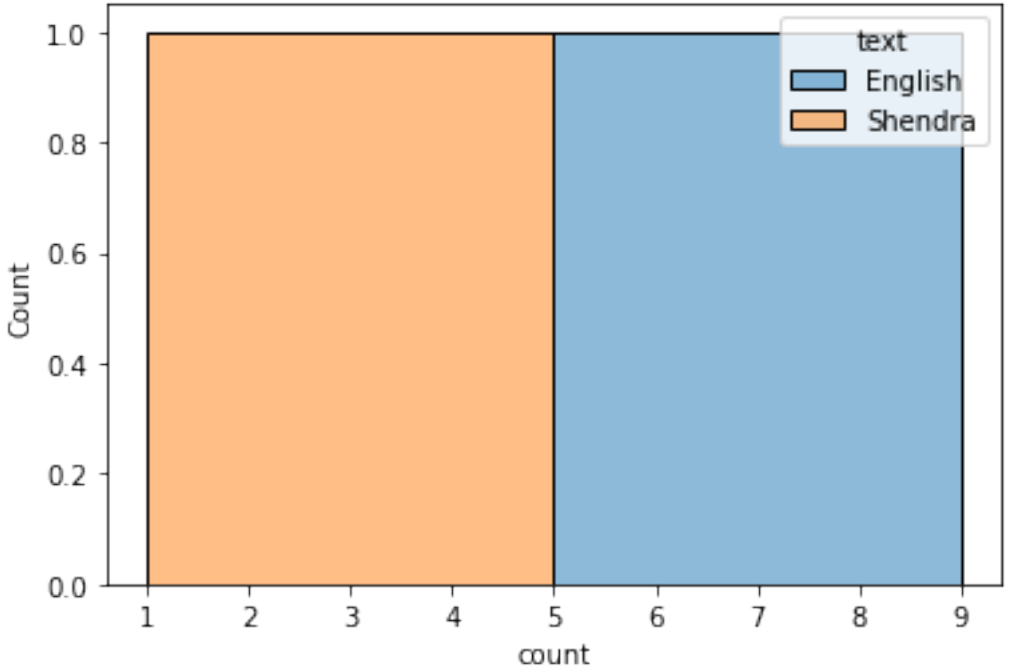
We can observe that most of people worked for GPE which showed a high distribution and next come CARDINAL and MICROSOFT

GPE Analysis



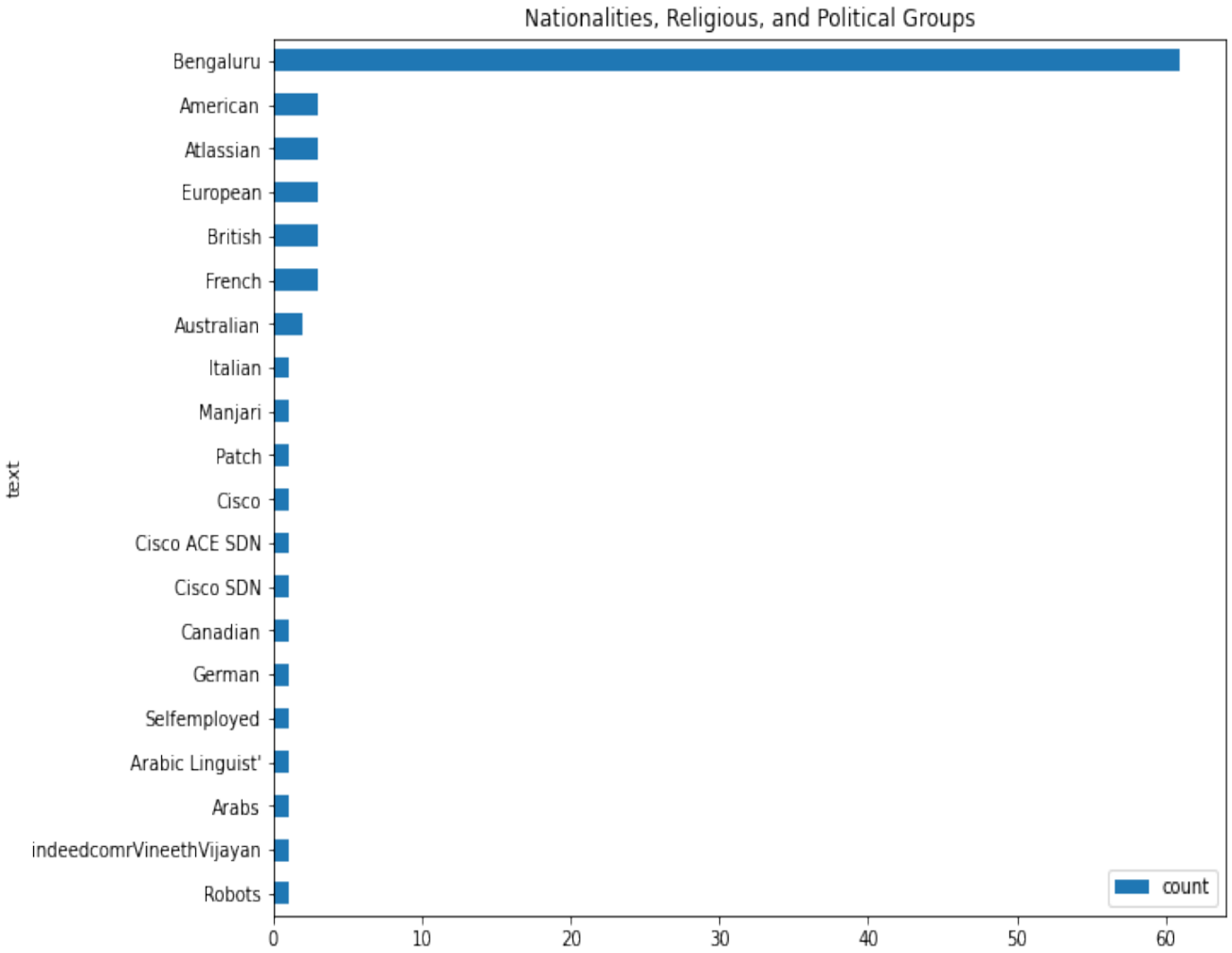
We can observe that the Indian country appear the most in the data which means that the resumes have been mostly submitted by people who leave in Indian. Then come US, UK and Mumbai.

LANGUAGE Analysis



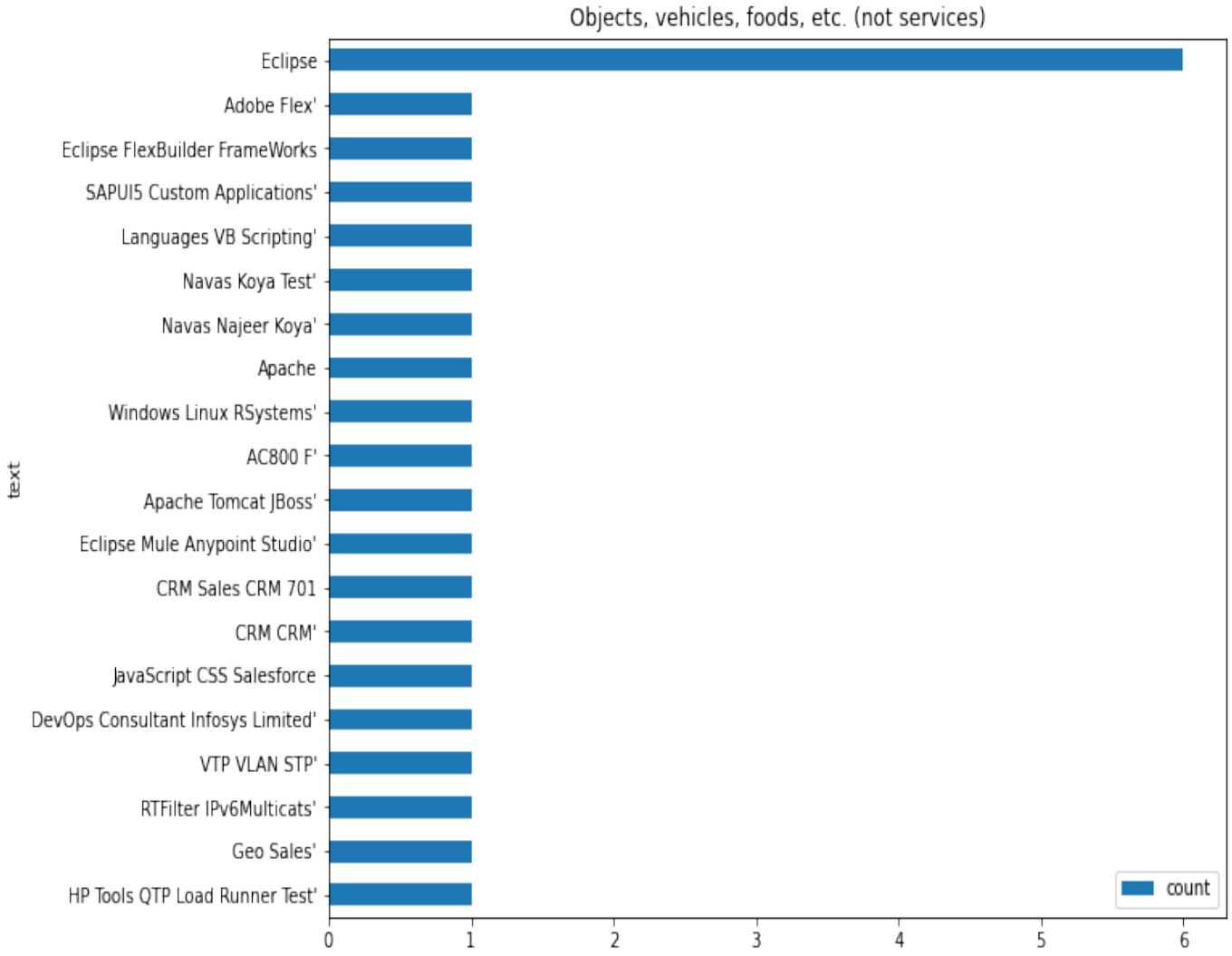
We can observe an equal distribution between the two languages (English and Shendra) found in the data which means that the candidates speak both English and Shendra

NORP Analysis



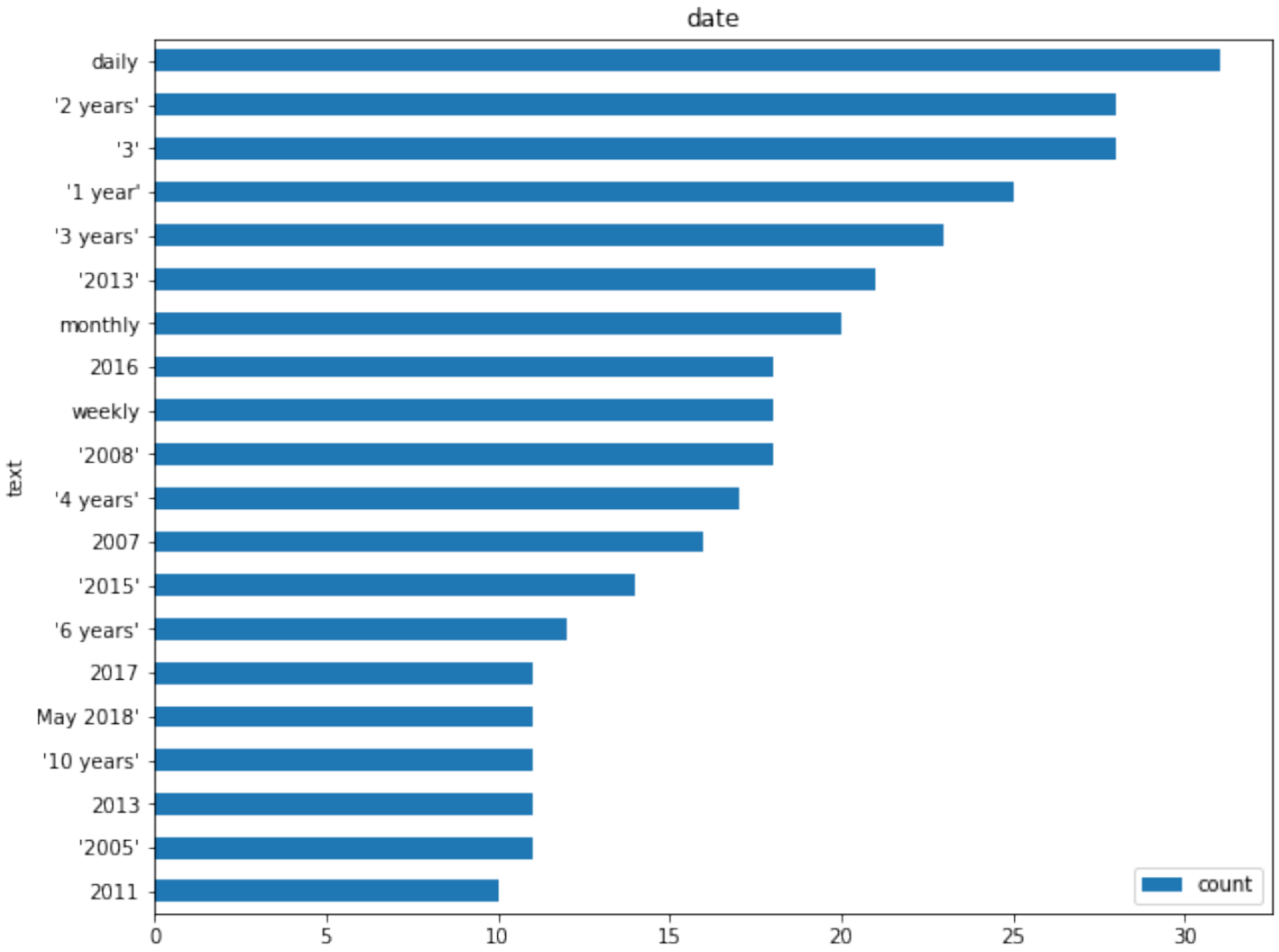
We can observe a higher distribution on Bengaluru which means that most of candidates come from Bengaluru. Then come American, Atlassian, European, British and French people.

PRODUCT Analysis



We can observe that many candidates have used Eclipse as material in their past work

Numbers of experience Analysis



We can observe that most of the candidates have done daily works and then most of them has 2-3 years of experiences in their fields

Recommended models

based on these observations, we recommend to focus on:

Years of experience : the years of experience of a candidate will give a certain idea of his profile and will testify to his skills and aptitudes to be shortlisted.

- **Companies:** The companies where they worked at will also give a considerable advantage to be shortlisted

- **Materials:** the materials employed by each candidate in his past work will determine wether the candidate fit the position or not.

On the analysis of above point and the given datasets , we will recommend to use a custom NER model in Spacy.
To do so we will need to:

- Install Spacy and Spcay transformers
- Have the text with the corresponding annotations

Model training

We first need to install spacy and spacy-transformers

- After having a look at the dataset, we only need text string, the entity start and end indices, and the entity type. S we then loop over our sentences and extract only those features
- We split our data into train/dev set and we load the model using spacy
- As spacy used DocBin class for annotated data, we'll create the DocBin objects for our training examples
- We now need to clean our dataset by creating a function that will remove leading and trailing white spaces from entity spans
- There are sone entity span overlaps such indices of some entities overlap, so we use the utility method filter_spans from spacy to deal with that
- We create and save a collection of training docs
- We create the config file generated using the quickstart page

Model training

Now we have all that we need to train our model. Using a CPU, the execution took around 1402.091s

```
2022-12-07 06:06:04.947774: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] fai
ℹ Saving to output directory: output
ℹ Using CPU

========================== Initializing pipeline ==========================
[2022-12-07 06:06:06,222] [INFO] Set up nlp object from config
INFO:spacy:Set up nlp object from config
[2022-12-07 06:06:06,234] [INFO] Pipeline: ['tok2vec', 'ner']
INFO:spacy:Pipeline: ['tok2vec', 'ner']
[2022-12-07 06:06:06,239] [INFO] Created vocabulary
INFO:spacy:Created vocabulary
[2022-12-07 06:06:06,240] [INFO] Finished initializing nlp object
INFO:spacy:Finished initializing nlp object
[2022-12-07 06:06:07,845] [INFO] Initialized pipeline components: ['tok2vec', 'ner']
INFO:spacy:Initialized pipeline components: ['tok2vec', 'ner']
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00    641.74    0.00    0.00    0.00    0.00
  1     200       2976.01   7022.81   19.82   63.10   11.75    0.20
  2     400      12721.43   5261.60   18.45   41.61   11.85    0.18
  4     600      13404.79   6513.68   26.67   47.57   18.53    0.27
  5     800      24534.65   3575.11   27.47   40.00   20.92    0.27
  7    1000      10350.73   2519.00   44.01   59.39   34.96    0.44
  8    1200      47692.60   2658.26   42.88   59.68   33.47    0.43
  9    1400      12574.60   2087.91   48.94   54.17   44.62    0.49
 11    1600       2211.42   1647.70   48.66   49.54   47.81    0.49
 12    1800       1449.99   1576.51   58.13   69.04   50.20    0.58
 14    2000       5580.34   1649.81   61.74   69.90   55.28    0.62
 15    2200      31086.84   1505.38   58.25   62.05   54.88    0.58
 17    2400       3094.48   1189.46   66.37   75.90   58.96    0.66
 18    2600      11359.91   1250.56   65.45   68.52   62.65    0.65
 20    2800       1746.20   1071.78   68.23   79.84   59.56    0.68
 21    3000      23893.26   1084.53   69.89   76.91   64.04    0.70
```

```
 21    3000      23893.26   1084.53   69.89   76.91   64.04    0.70
 23    3200       1771.24    970.81   69.58   80.68   61.16    0.70
 24    3400       1778.52    908.61   69.44   81.20   60.66    0.69
 26    3600      35932.66   1029.36   71.96   80.69   64.94    0.72
 27    3800      34622.33    933.81   73.19   82.24   65.94    0.73
 29    4000       1252.46    739.18   71.79   77.95   66.53    0.72
 31    4200       2089.11    790.02   71.35   78.01   65.74    0.71
 33    4400       1757.13    748.56   73.69   82.57   66.53    0.74
 35    4600     301282.42    933.83   75.03   85.68   66.73    0.75
 37    4800       1536.49    801.94   75.03   85.68   66.73    0.75
 39    5000       8501.81    813.18   74.42   82.89   67.53    0.74
 42    5200       2916.71    862.06   74.73   82.02   68.63    0.75
 45    5400       3642.62    905.17   74.08   78.64   70.02    0.74
 48    5600       2078.80    777.24   76.10   85.10   68.82    0.76
 50    5800       3090.95    803.76   75.97   85.54   68.33    0.76
 53    6000      17152.21    807.21   75.53   83.39   69.02    0.76
 56    6200       1536.64    701.80   76.14   84.74   69.12    0.76
 59    6400      64000.08    778.11   75.57   86.11   67.33    0.76
 62    6600       2208.16    654.42   75.59   83.68   68.92    0.76
 65    6800       1774.78    637.42   77.70   88.23   69.42    0.78
 68    7000       3319.50    669.51   74.67   80.37   69.72    0.75
 71    7200       2469.58    579.78   77.01   86.00   69.72    0.77
 74    7400       2582.14    637.46   75.15   86.01   66.73    0.75
 76    7600       2491.97    580.64   76.54   87.66   67.93    0.77
 79    7800       5937.82    630.06   75.91   83.16   69.82    0.76
 82    8000       4279.97    552.25   77.05   88.50   68.23    0.77
 85    8200       3159.91    549.71   75.76   85.48   68.03    0.76
 88    8400       3408.71    512.76   75.66   84.15   68.73    0.76
✔ Saved pipeline to output directory
output/model-last
```

Results

After training, we can load the best-performing model and test it on a piece of text as showed on the image below

```
nlp_ner = spacy.load("output/model-best")

doc = nlp_ner("dilliraja baskaran tamil nadu email indeed indeedcomrdillirajabaskaran4a3bc8a35879ce5c work experienc


spacy.displacy.render(doc, style="ent", jupyter=True)
```

dilliraja PERSON baskaran tamil nadu email indeed indeedcomrdillirajabaskaran4a3bc8a35879ce5c work GPE experience working infosys developer support infosys may 2015 present experience development project support project education panimalar engineering college skill mainframecoboljcldb2imsdbimsdc httpswwwindeedcomrdillirajabaskaran4a3bc8a35879ce5cisidrexdownloadikwdownloadtopcoin

Thank You