Problem Statement

While researching, she found a raw dataset of over 30k datapoints (Raw_Skills_Dataset.csv) which contain technical skills and a lot of jargon mixed in. Can you help her develop a code that can clean this dataset and extract Technical (Hard) skills?

Step-1 : Converted the Excel Files into Text files.

Step-2:

```
In [1]: #importing the dependencies
        import spacy
        from spacy.tokens import DocBin
        from tqdm import tqdm
        import json

        nlp = spacy.blank("en") # load a new spacy model
        db = DocBin() # create a DocBin object
```

Step-3: Gathering the training data by using [NER Annotator for SpaCy (tecoholic.github.io)]

I have used the "**Example_Technical_Skills**" file as Train Data and saved it a JSON file.

Step-4:

```
In [2]: #Loading the training data
        with open('annotations.json') as fp:
            train_data = json.load(fp)
```

Step-5: Looking at the train data

```
In [3]: train_data
```

```
Out[3]: {'classes': ['TECHNICAL SKILLS'],
         'annotations': [['SAP Fiori Developer\r',
           {'entities': [[0, 3, 'TECHNICAL SKILLS'], [10, 19, 'TECHNICAL SKILLS']]}],
          ['Oracle Instance Management & Strategy\r',
           {'entities': [[0, 6, 'TECHNICAL SKILLS'], [16, 26, 'TECHNICAL SKILLS']]}],
          ['Boomi Master Data Management\r',
           {'entities': [[0, 5, 'TECHNICAL SKILLS'],
             [6, 17, 'TECHNICAL SKILLS'],
             [18, 28, 'TECHNICAL SKILLS']]}],
          ['Digital Manufacturing on Cloud ( DMC)\r',
           {'entities': [[0, 30, 'TECHNICAL SKILLS']]}],
          ['DevOps\r', {'entities': [[0, 6, 'TECHNICAL SKILLS']]}],
          ['CA SAM\r',
           {'entities': [[0, 2, 'TECHNICAL SKILLS'], [3, 6, 'TECHNICAL SKILLS']]}],
          ['OpenShift\r', {'entities': [[0, 9, 'TECHNICAL SKILLS']]}],
          ['Acxiom Data Analytics\r', {'entities': [[7, 21, 'TECHNICAL SKILLS']]}],
          ['SAP Digital Boardroom\r',
           {'entities': [[0, 3, 'TECHNICAL SKILLS'],
             [4, 11, 'TECHNICAL SKILLS'],
```

Step-6: Checking the version of spaCY being used

```
In [4]: !python -m spacy info

        ============================== Info about spaCy ==============================

        spaCy version    3.3.0
        Location         C:\Python310\lib\site-packages\spacy
        Platform         Windows-10-10.0.19044-SP0
        Python version   3.10.4
        Pipelines        en_core_web_sm (3.3.0)
```

Step-7: Converting the train data to a format understandable by spaCY that is .spacy format

In [7]:
```
for text, annot in tqdm(train_data['annotations']):
    doc = nlp.make_doc(text)
    ents = []
    for start, end, label in annot["entities"]:
        span = doc.char_span(start, end, label=label, alignment_mode="contract")
        if span is None:
            print("Skipping entity")
        else:
            ents.append(span)
    doc.ents = ents
    db.add(doc)

db.to_disk("./training_data.spacy") # save the docbin object
```

```
100%|██████████| 911/911 [00:00<00:00, 2614.40it/s]
Skipping entity
Skipping entity
```

Step-8: Running the training configuration

In [6]:
```
! python -m spacy train config.cfg --output ./ --paths.train ./training_data.spacy --paths.dev ./training_data.spacy
```

```
[i] Saving to output directory: .
[2022-05-28 14:01:47,159] [INFO] Set up nlp object from config
[2022-05-28 14:01:47,166] [INFO] Pipeline: ['tok2vec', 'ner']
[2022-05-28 14:01:47,168] [INFO] Created vocabulary
[2022-05-28 14:01:47,171] [INFO] Finished initializing nlp object
[2022-05-28 14:01:47,448] [INFO] Initialized pipeline components: ['tok2vec', 'ner']
[i] Using CPU

============================ Initializing pipeline ============================
[+] 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     68.83   33.22   23.15   58.78    0.33
  5     200        245.39   4986.53   87.44   88.07   86.81    0.87
 11     400        345.59   1804.85   95.34   95.88   94.80    0.95
 18     600        312.82   1316.61   96.34   96.38   96.31    0.96
 27     800        279.41   1197.11   96.25   96.66   95.86    0.96
 39    1000        225.87   1160.77   96.62   97.32   95.93    0.97
 52    1200        145.16   1133.54   96.70   97.40   96.01    0.97
 69    1400        175.14   1324.15   96.78   97.33   96.23    0.97
 89    1600        135.76   1483.63   96.79   97.05   96.53    0.97
113    1800        118.96   1643.10   96.70   97.33   96.08    0.97
143    2000         85.71   1924.91   96.85   97.48   96.23    0.97
180    2200         90.43   2269.85   96.86   97.19   96.53    0.97
220    2400         90.71   2492.49   96.74   96.34   97.14    0.97
260    2600         89.58   2454.87   96.85   96.28   97.44    0.97
300    2800         82.49   2448.42   96.57   96.61   96.53    0.97
340    3000         69.53   2420.65   96.52   97.91   95.18    0.97
380    3200         75.25   2410.76   96.65   96.54   96.76    0.97
420    3400         65.89   2410.49   96.76   97.92   95.63    0.97
460    3600         75.90   2402.31   96.79   96.90   96.68    0.97
500    3800         59.16   2403.94   96.84   97.85   95.86    0.97
[+] Saved pipeline to output directory
model-last
```

Step-9: Loading the best model

:
```
nlp_ner = spacy.load("C:/Users/Personal/OneDrive/Documents/Jupyter Notebook/model-best")
```

Step-10: Giving the test data as input

```
doc = nlp_ner('''RAW DATA
What ifs
seniority
familiarity
functionalities
Lambdas
Java Streams
Object Oriented analysis
Relational Databases
SQL
ORM
JPA2
Hibernate
MyBatis
Hibernate
code versioning tools
Git.. Familiarity
Maven
Gradle.. Familiarity
continuous integration
continuous delivery development processes
jenkins
bamboo
```

Step-11: Displaying the technical skills in jupyter

```
In [9]: spacy.displacy.render(doc, style="ent", jupyter=True) # display in Jupyter
```

toolsprofiling tools Bachelor or **TECHNICAL SKILLS** Master degree STEM majors **TECHNICAL SKILLS** Strong algorithmscoding backgroundeither Java **TECHNICAL SKILLS** PythonScala programming **TECHNICAL SKILLS** experienceExceptional proficiency SQL **TECHNICAL SKILLS** each following distributed technologies Relational Storesi.e. Postgres MySQL **TECHNICAL SKILLS** Oracle **TECHNICAL SKILLS** NoSQL **TECHNICAL SKILLS** Stores **TECHNICAL SKILLS** (i.e. Big Query **TECHNICAL SKILLS** Clickhouse Distribut Processing Engines **TECHNICAL SKILLS** i.e. Apache Spark Apache **TECHNICAL SKILLS** FlinkCeleryDistributed Queuesi.e. Apache Kafka **TECHNICAL SKILLS** AWS **TECHNICAL** Kinesis **TECHNICAL SKILLS** GCP PusSub standard methodologies **TECHNICAL SKILLS** GCP Azure **TECHNICAL SKILLS** AWS **TECHNICAL SKILLS** similar cloud platform techno persuasion **TECHNICAL SKILLS** experience building cloud environments Excellent Unix/Linux **TECHNICAL SKILLS** experience and programming **TECHNICAL SKILLS** experi scripting "python, **TECHNICAL SKILLS** shell"IPC mechanismsTCP/IPKafkazmqgrpc microservice architecture **TECHNICAL SKILLS** orchestration **TECHNICAL SKILLS** docker kubernetes **TECHNICAL SKILLS** cloud environments **TECHNICAL SKILLS** etcdservice meshesenvoyvault API gateways **TECHNICAL SKILLS** load balancersdatabasesconsistencysql/nosql columnar databases **TECHNICAL SKILLS** distributed consensus CAP Theorem RAFT "map **TECHNICAL SKILLS** -reduce, reliable multicast" **TECHNICAL SKILLS** PaxosCI/CD Github **TECHNICAL SKILLS** Makefiles Microservices **TECHNICAL SKILLS** AWS **TECHNICAL SKILLS** GCPNode.js REST APIs servicesExperience writing tools **TECHNICAL SKILLS** Ruby PerlExperience **TECHNICAL SKILLS** Bazel **TECHNICAL SKILLS** toolingScala programming **TECHNICAL SKILLS**