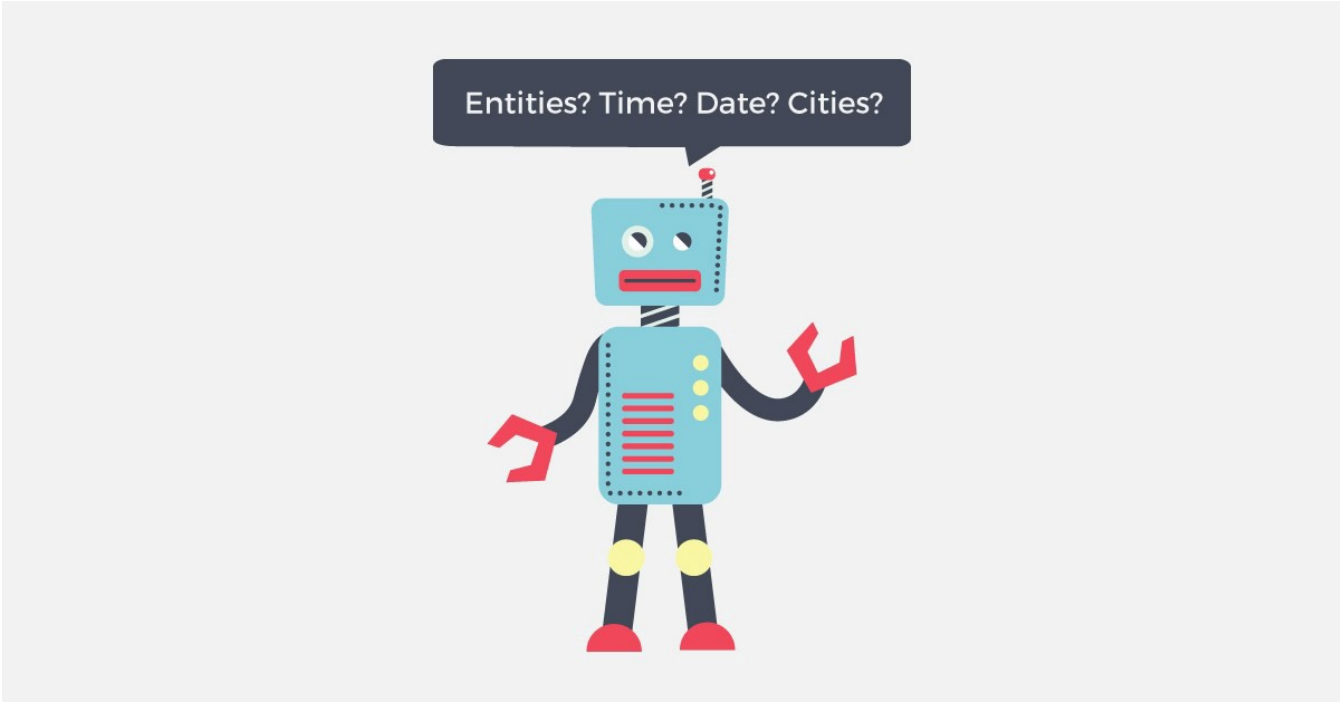
A Review of Named Entity Recognition (NER) Using Automatic Summarization of Resumes

Mohan Gupta
Jul 10, 2018 · 11 min read

Understand what NER is and how it is used in the industry, various libraries for NER, code walk through of using NER for resume summarization.



This blog speaks about a field in Natural language Processing (NLP) and Information Retrieval (IR) called Named Entity Recognition and how we can apply it for automatically generating summaries of resumes by extracting only chief entities like name, education background, skills, etc.

What is Named Entity Recognition?

Named-entity recognition (**NER**) (also known as **entity identification**, **entity chunking** and **entity extraction**) is a sub-task of information extraction that seeks to locate and classify named entities in text into pre-defined categories such as the names

of persons, organizations, locations, expressions of times, quantities, monetary values, percentages, etc.

NER systems have been created that use linguistic grammar-based techniques as well as statistical models such as machine learning. Hand-crafted grammar-based systems typically obtain better precision, but at the cost of lower recall and months of work by experienced computational linguists . Statistical NER systems typically require a large amount of manually annotated training data. Semi-supervised approaches have been suggested to avoid part of the annotation effort.

State-of-the-Art NER Models

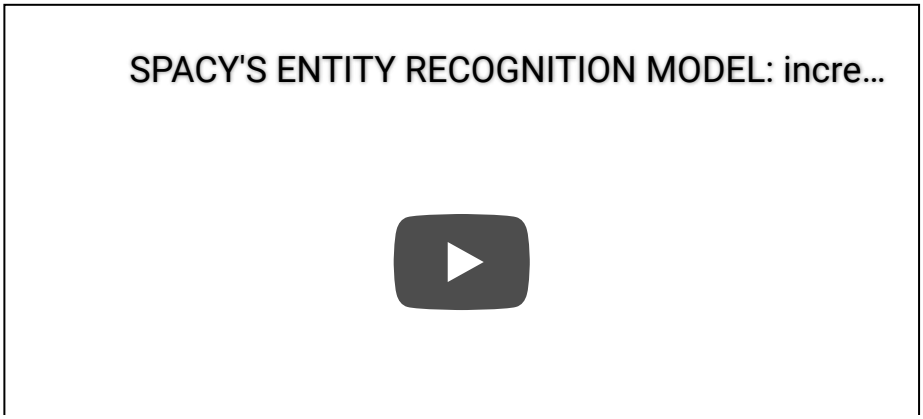
spaCy NER Model :

Being a free and an open-source library, spaCy has made advanced Natural Language Processing (NLP) much simpler in Python.

spaCy provides an exceptionally efficient statistical system for named entity recognition in python, which can assign labels to groups of tokens which are contiguous. It provides a default model which can recognize a wide range of named or numerical entities, which include company-name, location, organization, product-name, etc to name a few. Apart from these default entities, spaCy enables the addition of arbitrary classes to the entity-recognition model, by training the model to update it with newer trained examples.

Model Architecture :

The statistical models in spaCy are custom-designed and provide an exceptional performance mixture of both speed, as well as accuracy. The current architecture used has not been published yet, but the following video gives an overview as to how the model works with primary focus on NER model.



Stanford Named Entity Recognizer :

Stanford NER is a Named Entity Recognizer, implemented in Java. It provides a default trained model for recognizing chiefly entities like Organization, Person and Location. Apart from this, various models trained for different languages and circumstances are also available.

Model Architecture :

Stanford NER is also referred to as a CRF (Conditional Random Field) Classifier as Linear chain Conditional Random Field (CRF) sequence models have been implemented in the software. We can train our own custom models with our own labeled dataset for various applications.

CRF models were originally pioneered by Lafferty, McCallum, and Pereira (2001); Please refer to Sutton and McCallum (2006) or Sutton and McCallum (2010) for detailed comprehensible introductions.

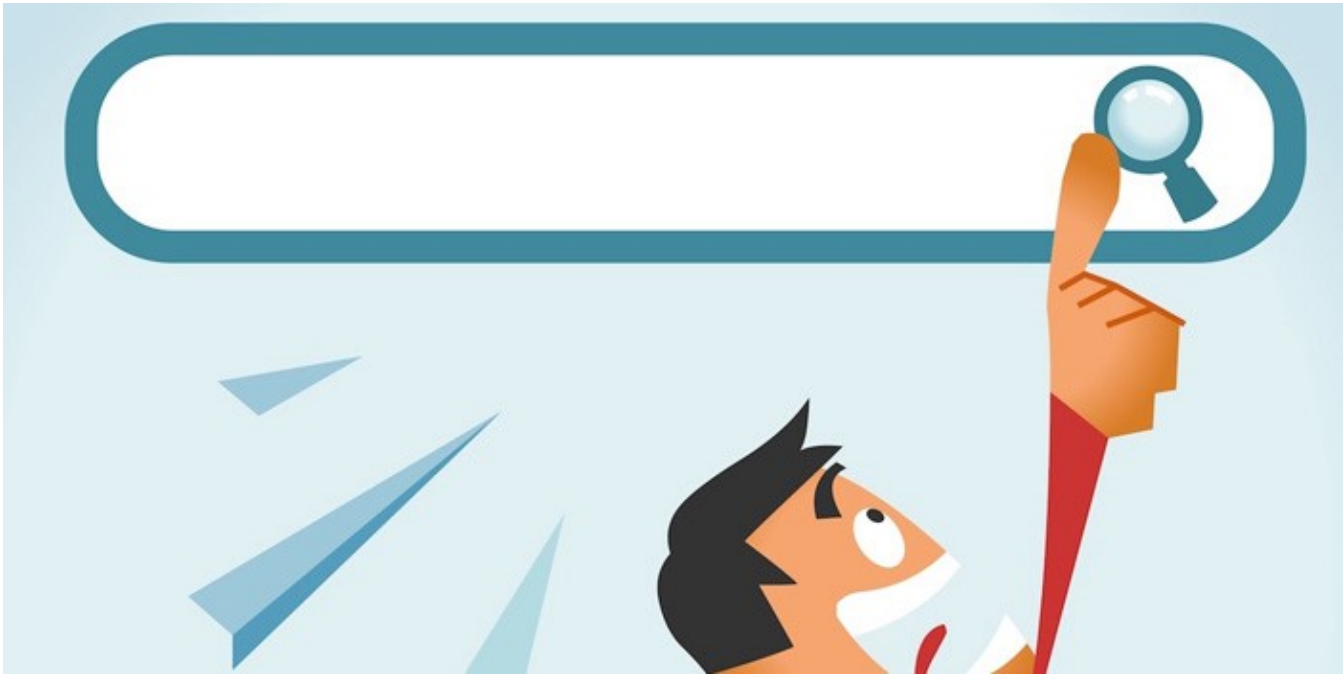
Use Cases of NER Models

Named Entity Recognition has a wide range of applications in the field of Natural Language Processing and Information Retrieval. Few such examples have been listed below :

Automatically Summarizing Resumes :

One of the key challenges faced by the HR Department across companies is to evaluate a gigantic pile of resumes to shortlist candidates. To add to their burden, resumes of applicants are often excessively populated in detail, of which, most of the information is irrelevant to what the evaluator is seeking. With the aim of simplifying this process, through our NER model, we could facilitate evaluation of resumes at a quick glance, thereby simplifying the effort required in shortlisting candidates among a pile of resumes.

Optimizing Search Engine Algorithms :



To design a search engine algorithm, instead of searching for an entered query across the millions of articles and websites online, a more efficient approach would be to run an NER model on the articles once and store the entities associated with them permanently. The key tags in the search query can then be compared with the tags associated with the website articles for a quick and efficient search.

Powering Recommender Systems :

NER can be used in developing algorithms for recommender systems which automatically filter relevant content we might be interested in and accordingly guide us to discover related and unvisited relevant contents based on our previous behaviour. This may be achieved by extracting the entities associated with the content in our history or previous activity and comparing them with label assigned to other unseen content to filter relevant ones.

Simplifying Customer Support :

NER can be used in recognizing relevant entities in customer complaints and feedback such as Product specifications, department or company branch details, so that the feedback is classified accordingly and forwarded to the appropriate department responsible for the identified product.

We describe summarization of resumes using NER models in detail in the further sections.

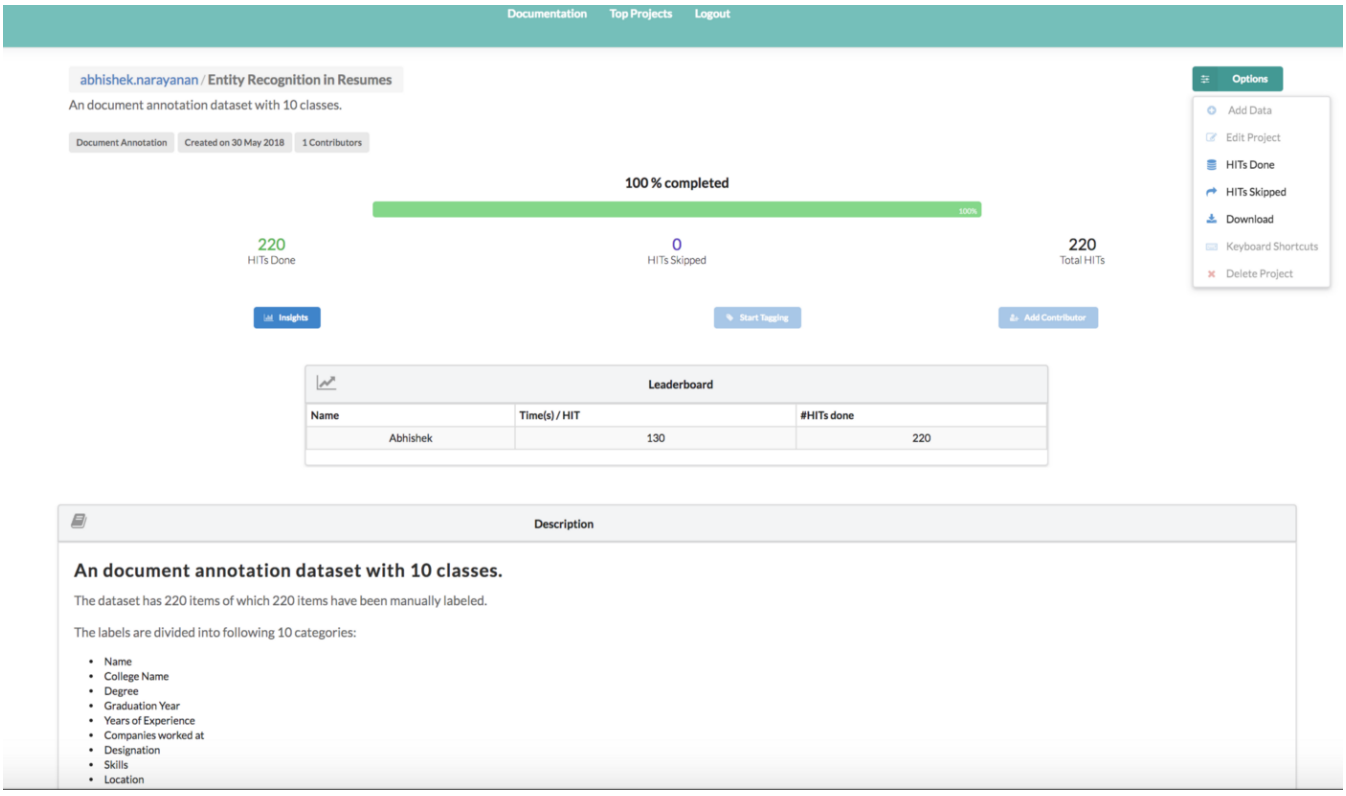
NER For Resume Summarization

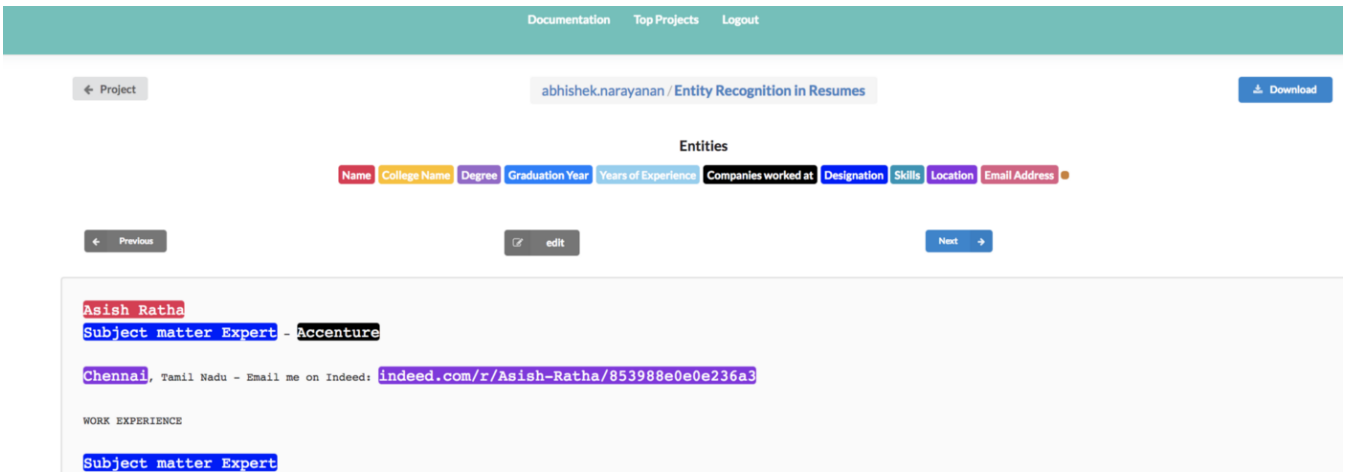
Dataset :

The first task at hand of course is to create manually annotated training data to train the model. For this purpose, 220 resumes were downloaded from an online jobs platform. These documents were uploaded to Dataturks online annotation tool and manually annotated.

The tool automatically parses the documents and allows for us to create annotations of important entities we are interested in and generates JSON formatted training data with each line containing the text corpus along with the annotations.

A snapshot of the dataset can be seen below :





The above dataset consisting of 220 annotated resumes can be found here. We train the model with 200 resume data and test it on 20 resume data.

Using spaCy model in python for training a custom model :

Dataset format :

A sample of the generated json formatted data generated by the Dataturks annotation tool, which is supplied to the code is as follows :


```
[
  {
    "label": [
      "Email Address"
    ],
    "points": [
      {
        "start": 998,
        "end": 1037,
        "text": "indeed.com/r/Pavithra-M/26f392ec8251143b"
      }
    ]
  },
  {
    "label": [
      "Skills"
    ],
    "points": [
      {
        "start": 611,
        "end": 983,
        "text": "ADOBE PHOTOSHOP (Less than 1 year), ANDROID (Less than 1 year), APPLICA
          :\n\n• Programing Languages: C, C++ and JAVA.\n\n• Databases: MySQL, SQL serve
      }
    ]
  },
  {
    "label": [
      "Designation"
    ],
    "points": [
      {
        "start": 432,
        "end": 441,
        "text": "internship"
      }
    ]
  },
```

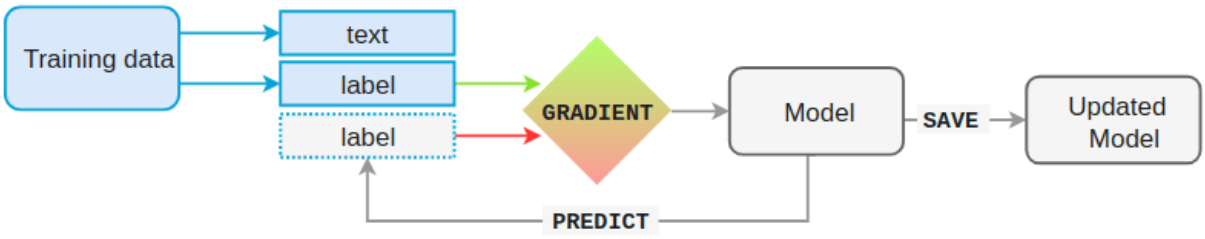
```
{
  "label": [
    "Companies worked at"
  ],
  "points": [
    {
      "start": 423,
      "end": 429,
      "text": "Infosys"
    }
  ]
```


Training the Model :

We use python's spaCy module for training the NER model. spaCy's models are **statistical** and every "decision" they make — for example, which part-of-speech tag to assign, or whether a word is a named entity — is a **prediction**. This prediction is based on the examples the model has seen during **training**.

The model is then shown the unlabelled text and will make a prediction. Because we know the correct answer, we can give the model feedback on its prediction in the form of an **error gradient** of the **loss function** that calculates the difference between the training example and the expected output. The greater the difference, the more significant the gradient and the updates to our model.

When training a model, we don't just want it to memorise our examples — we want it to come up with theory that can be **generalised across other examples**. After all, we don't just want the model to learn that this one instance of "Amazon" right here is a company — we want it to learn that "Amazon", in contexts *like this*, is most likely a company. In order to tune the accuracy, we process our training examples in batches, and experiment with `minibatch` sizes and dropout rates.



Of course, it's not enough to only show a model a single example once. Especially if you only have few examples, you'll want to train for a **number of iterations**. At each iteration, the training data is **shuffled** to ensure the model doesn't make any generalisations based on the order of examples.

Another technique to improve the learning results is to set a **dropout rate**, a rate at which to randomly "drop" individual features and representations. This makes it harder for the model to memorise the training data. For example, a `0.25` dropout means that each feature or internal representation has a 1/4 likelihood of being dropped. We train the model for 10 epochs and keep the dropout rate as 0.2.

Here's a code snippet for training the model :

```
import spacy
################### Train Spacy NER.############
def train_spacy():

    TRAIN_DATA = convert_dataturks_to_spacy("/home/abhishekn/dataturks/entityrecognition/trainc
    nlp = spacy.blank('en')  # create blank Language class
    # create the built-in pipeline components and add them to the pipeline
    # nlp.create_pipe works for built-ins that are registered with spaCy
    if 'ner' not in nlp.pipe_names:
        ner = nlp.create_pipe('ner')
        nlp.add_pipe(ner, last=True)


    # add labels
    for _, annotations in TRAIN_DATA:
         for ent in annotations.get('entities'):
            ner.add_label(ent[2])

    # get names of other pipes to disable them during training
    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("Statring iteration " + str(itn))
            random.shuffle(TRAIN_DATA)
            losses = {}
            for text, annotations in TRAIN_DATA:
                nlp.update(
                    [text],  # batch of texts
                    [annotations],  # batch of annotations
                    drop=0.2,  # dropout - make it harder to memorise data
                    sgd=optimizer,  # callable to update weights
                    losses=losses)
            print(losses)
```

Contribute to Entity-Recognition-In-Resumes-SpaCy development by creating an account on GitHub.

github.com

Results and Evaluation of the spaCy model :

The model is tested on 20 resumes and the predicted summarized resumes are stored as separate .txt files for each resume.

For each resume on which the model is tested, we calculate the accuracy score, precision, recall and f-score for each entity that the model recognizes. The values of these metrics for each entity are summed up and averaged to generate an overall score to evaluate the model on the test data consisting of 20 resumes. The entity wise evaluation results can be observed below . It is observed that the results obtained have been predicted with a commendable accuracy.

Recognized Entity	Precision	Recall	F-Score
College Name	100%	100%	100%
Location	99.28%	99.27%	99.27%
Designation	100%	98.785	99.395
Email Address	100%	99.43%	99.71%
Name	97.83%	97.83%	97.83%
Skills	94.30%	98.40%	96.32%

A sample summary of an unseen resume of an employee from indeed.com obtained by prediction by our model is shown below :

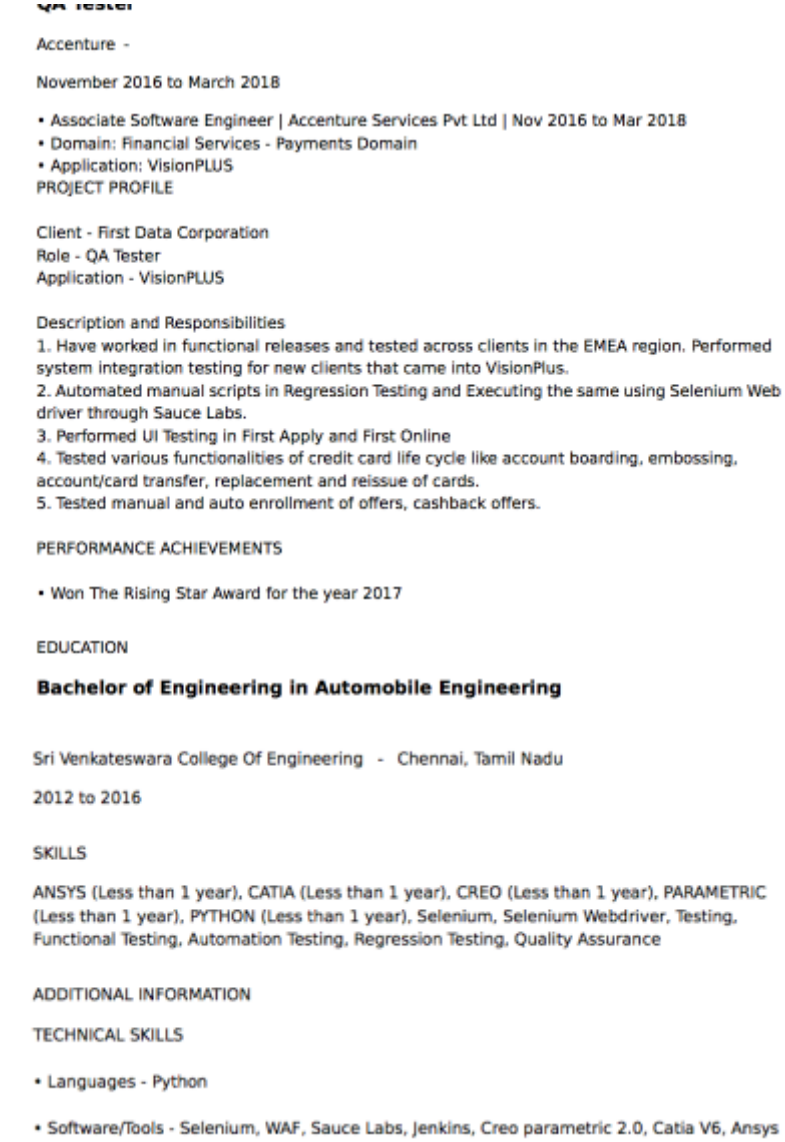
Arun Elumalai
QA Tester

Chennai, Tamil Nadu - Email me on Indeed: indeed.com/r/Arun-Elumalai/26575d617d50ea04

- 15 Months of Experience as a QA Tester in Software Testing (Mainframe)
- Experience in Automation, Functional, UI testing and Regression Testing.
- Involvement in preparation of Test scenarios, Test cases and executing the same.
- Defect reporting and tracking via Rational Quality Manager.
- Preparation of test closure reports.

WORK EXPERIENCE

QA Tester



QA Tester

Accenture -

November 2016 to March 2018

• Associate Software Engineer | Accenture Services Pvt Ltd | Nov 2016 to Mar 2018
• Domain: Financial Services - Payments Domain
• Application: VisionPLUS
PROJECT PROFILE

Client - First Data Corporation
Role - QA Tester
Application - VisionPLUS

Description and Responsibilities
1. Have worked in functional releases and tested across clients in the EMEA region. Performed system integration testing for new clients that came into VisionPlus.
2. Automated manual scripts in Regression Testing and Executing the same using Selenium Web driver through Sauce Labs.
3. Performed UI Testing in First Apply and First Online
4. Tested various functionalities of credit card life cycle like account boarding, embossing, account/card transfer, replacement and reissue of cards.
5. Tested manual and auto enrollment of offers, cashback offers.

PERFORMANCE ACHIEVEMENTS

• Won The Rising Star Award for the year 2017

EDUCATION

Bachelor of Engineering in Automobile Engineering

Sri Venkateswara College Of Engineering - Chennai, Tamil Nadu

2012 to 2016

SKILLS

ANSYS (Less than 1 year), CATIA (Less than 1 year), CREO (Less than 1 year), PARAMETRIC (Less than 1 year), PYTHON (Less than 1 year), Selenium, Selenium Webdriver, Testing, Functional Testing, Automation Testing, Regression Testing, Quality Assurance

ADDITIONAL INFORMATION

TECHNICAL SKILLS

• Languages - Python

• Software/Tools - Selenium, WAF, Sauce Labs, Jenkins, Creo parametric 2.0, Catia V6, Ansys

Resume of an Accenture employee obtained from indeed.com

```
Degree:
Bachelor of Engineering in Automobile Engineering


Designation:
QA Tester


Graduation Year:
2016


Name:
Arun Elumalai


Companies worked at:
Accenture


Email Address:
indeed.com/r/Arun-Elumalai/26575d617d50ea04


Location:
Chennai
```

```
Skills:
ANSYS (Less than 1 year), CATIA (Less than 1 year), CREO (Less than 1 year),
PARAMETRIC(Less than 1 year), PYTHON (Less than 1 year), Selenium, Selenium Webdriver,
Testing,Functional Testing, Automation Testing, Regression Testing, Quality Assurance
```

Summarized Resume as obtained in output

Using Stanford NER model in Java for training a custom model :

Dataset Format :

The data for training has to be passed as a text file such that every line contains a word-label pair, where the word and the label tag are separated by a tab space '\t'. For a text document,as in our case, we tokenize documents into words and add one line for each word and associated tag into the training file. To indicate the start of the next file, we add an empty line in the training file.

Here is a sample of the input training file:

```
Abhishek    Name
Jha Name
Application Designation
Development Designation
Associate   Designation
-   O
Accenture   Companies worked at
Bengaluru   Location
,   O
Karnataka   O
-   O
Email   O
me  O
on  O
Indeed: Email Address
indeed.com/r/Abhishek-Jha/10e7a8cb732bc43a  Email Address
```

Note: It is compulsory to include a label/tag for each word. Here, for words we do not care about we are using the label zero '0'.

Properties file :

Stanford CoreNLP requires a properties file where the parameters necessary for building a custom model. For instance, we may define ways of extracting features for learning, etc. Following is an example of a properties file:

```
# location of the training file
trainFile = ./standford_train.txt
# location where you would like to save (serialize) your
# classifier; adding .gz at the end automatically gzips the file,
```

```
# making it smaller, and faster to load
serializeTo = ner-model.ser.gz

# structure of your training file; this tells the classifier that
# the word is in column 0 and the correct answer is in column 1
map = word=0,answer=1

# This specifies the order of the CRF: order 1 means that features
# apply at most to a class pair of previous class and current class
# or current class and next class.
maxLeft=1

# these are the features we'd like to train with
# some are discussed below, the rest can be
# understood by looking at NERFeatureFactory
useClassFeature=true
useWord=true
# word character ngrams will be included up to length 6 as prefixes
# and suffixes only
useNGrams=true
noMidNGrams=true
maxNGramLeng=6
usePrev=true
useNext=true
useDisjunctive=true
useSequences=true
usePrevSequences=true
# the last 4 properties deal with word shape features
useTypeSeqs=true
useTypeSeqs2=true
useTypeySequences=true
#wordShape=chris2useLC
wordShape=none
#useBoundarySequences=true
#useNeighborNGrams=true
#useTaggySequences=true
#printFeatures=true
#saveFeatureIndexToDisk = true
#useObservedSequencesOnly = true
#useWordPairs = true
```

Training the model :

The chief class in Stanford CoreNLP is `CRFClassifier`, which possesses the actual model. In the code provided in the Github repository, the link to which has been attached below, we have provided the code to train the model using the training data and the properties file and save the model to disk to avoid time consumption for training each time. Next time we use the model for prediction on an unseen document, we just load the trained model from disk and use to for classification.

The first column in the output contains the input tokens while the second column refers to the correct label, and the third column is the label predicted by the classifier.

Here's a Code snippet for training the model and saving it to disk:

```java
public void trainAndWrite(String modelOutPath, String prop, String trainingFilepath) {
        Properties props = StringUtils.propFileToProperties(prop);
        props.setProperty("serializeTo", modelOutPath);

        //if input use that, else use from properties file.
        if (trainingFilepath != null) {
            props.setProperty("trainFile", trainingFilepath);
        }


        SeqClassifierFlags flags = new SeqClassifierFlags(props);


        CRFClassifier<CoreLabel> crf = new CRFClassifier<>(flags);
        crf.train();

        crf.serializeClassifier(modelOutPath);
    }
```

train.java hosted with ♡ by GitHub view raw

DataTurks-Engg/Entity-Recognition-In-Resumes-StanfordNER

Contribute to Entity-Recognition-In-Resumes-StanfordNER development by creating an account on GitHub.

github.com

Results and Evaluation of the Stanford NER model :

The model is tested on 20 resumes and the predicted summarized resumes are stored as separate .txt files for each resume.

For each resume on which the model is tested, we calculate the accuracy score, precision, recall and f-score for each entity that the model recognizes. The values of these metrics for each entity are summed up and averaged to generate an overall score to evaluate the model on the test data consisting of 20 resumes. The entity wise

evaluation results can be observed below . It is observed that the results obtained have been predicted with a commendable accuracy.

Recognized Entity	Precision	Recall	F-Score
College Name	100.0%	100.0%	100%.0
Location	100.0%	97.78%	98.88%
Designation	100.0%	100.0%	100.0%
Email Address	95.83%	100.0%	97.87%
Name	100.0%	100.0%	100%.0
Skills	96.36%	96.36%	96.36%
Years of Experience	100.0%	100.0%	100.0%
Graduation Year	96.55%	87.50%	91.80%
Degree	100.0%	100.0%	100.0%
Companies worked at	98.08%	100.0%	99.03%

A sample summary of an unseen resume of an employee from indeed.com obtained by prediction by our model is shown below :

Arun Elumalai
QA Tester

Chennai, Tamil Nadu - Email me on Indeed: indeed.com/r/Arun-Elumalai/26575d617d50ea04

• 15 Months of Experience as a QA Tester in Software Testing (Mainframe)

• Experience in Automation, Functional, UI testing and Regression Testing.

• Involvement in preparation of Test scenarios, Test cases and executing the same.

• Defect reporting and tracking via Rational Quality Manager.

• Preparation of test closure reports.

WORK EXPERIENCE

QA Tester

Accenture -

November 2016 to March 2018

• Associate Software Engineer | Accenture Services Pvt Ltd | Nov 2016 to Mar 2018
• Domain: Financial Services - Payments Domain
• Application: VisionPLUS
PROJECT PROFILE

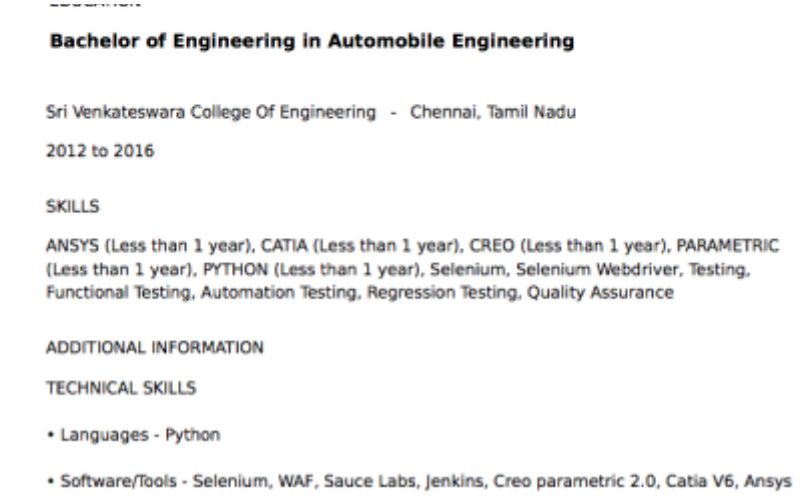
Client - First Data Corporation
Role - QA Tester
Application - VisionPLUS

Description and Responsibilities
1. Have worked in functional releases and tested across clients in the EMEA region. Performed system integration testing for new clients that came into VisionPlus.
2. Automated manual scripts in Regression Testing and Executing the same using Selenium Web driver through Sauce Labs.
3. Performed UI Testing in First Apply and First Online
4. Tested various functionalities of credit card life cycle like account boarding, embossing, account/card transfer, replacement and reissue of cards.
5. Tested manual and auto enrollment of offers, cashback offers.

PERFORMANCE ACHIEVEMENTS

• Won The Rising Star Award for the year 2017

EDUCATION



Bachelor of Engineering in Automobile Engineering

Sri Venkateswara College Of Engineering - Chennai, Tamil Nadu

2012 to 2016

SKILLS

ANSYS (Less than 1 year), CATIA (Less than 1 year), CREO (Less than 1 year), PARAMETRIC (Less than 1 year), PYTHON (Less than 1 year), Selenium, Selenium Webdriver, Testing, Functional Testing, Automation Testing, Regression Testing, Quality Assurance

ADDITIONAL INFORMATION

TECHNICAL SKILLS

• Languages - Python

• Software/Tools - Selenium, WAF, Sauce Labs, Jenkins, Creo parametric 2.0, Catia V6, Ansys

A resume of an Accenture employee obtained from indeed.com

Name:
Arun Elumalai

Degree:
Bachelor of Engineering in Automobile Engineering

Designation:
QA Tester

Skills:
UI testing and Regression cashback Selenium Functional Automation Regression • Languages - Python • - Sauce Creo parametric Catia

Summarized Resume as obtained in Output

Comparison of spaCy , Stanford NER and State-of-the-Art Models :

The vast majority of tokens in real-world resume documents are not part of entity names as usually defined, so the baseline precision, recall is extravagantly high, typically >90%; going by this logic, the entity wise precision recall values of both the models are reasonably good.

From the evaluation of the models and the observed outputs, spaCy seems to outperform Stanford NER for the task of summarizing resumes. A review of the F-scores for the entities identified by both models is as follows :

Recognized Entity	F-Score for spaCy	F-Score for Stanford NER

College Name	100.0%	100%.0
Location	98.97%	98.88%
Designation	99.39%	100.0%
Email Address	99.71%	97.87%
Name	99.81%	100%.0
Skills	100.0%	96.36%
MEAN	**99.64%**	**98.85%**

Here is the dataset of the resumes tagged with NER entities.

The Python code for the above project for training the spaCy model can be found here in the github repository.

The Java code for the above project for training the Stanford NER model can be found here in the GitHub repository.

Note: This blog is an extended version of the NER blog published at Dataturks.

Thanks to Hamza Bendemra, Ph.D..

Machine Learning Named Entity Recognition NLP

About Help Legal