

Comparison of NLP Tools on Named-Entity Recognition of Resumes

Artsiom Strok
NetID: *astrok2*
astrok2@illinois.edu

Abstract—Named entity recognition (NER) – is an information extraction technique that automatically identifies named entities in a text and classifies them into predefined categories such as person name, organization, location. In this article, I am going to compare open-source APIs SpaCy and NLTK (Stanford NER Tagger) as well as SaaS APIs Google Cloud Natural Language API, Amazon Comprehend Text Analysis API, Microsoft Azure Text Analytics API on Named-Entity Recognition in the text of resumes.

Index Terms—Resume, NER, SpaCy, NLTK, Stanford NER Tagger, Google Cloud Natural Language API, Amazon Comprehend Text Analysis API, Microsoft Azure Text Analytics API

I. INTRODUCTION

This section has short overview of each of the NLP tool with additional details regarding supported languages, named entity types, example of API, support of custom NER model training and additional notes.

A. SpaCy

SpaCy¹ is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. The library is published under the MIT license and its main developers are Matthew Honnibal and Ines Montani, the founders of the software company Explosion.

Unlike NLTK, which is widely used for teaching and research, spaCy focuses on providing software for production usage. As of version 1.0, spaCy also supports deep learning workflows that allow connecting statistical models trained by popular machine learning libraries like TensorFlow, PyTorch or MXNet through its own machine learning library Thinc. Using Thinc as its backend, spaCy features convolutional neural network models for part-of-speech tagging, dependency parsing, text categorization and named entity recognition (NER). Prebuilt statistical neural network models to perform these tasks are available for English, German, Greek, Spanish, Portuguese, French, Italian, Dutch, Lithuanian and Norwegian, and there is also a multi-language NER model. Additional support for tokenization for more than 50 languages allows users to train custom models on their own datasets as well.

Supported NER categories

- CARDINAL
- DATE

- EVENT
- FAC
- GPE
- LANGUAGE
- LAW
- LOC
- MONEY
- NORP
- ORDINAL
- ORG
- PERCENT
- PERSON
- PRODUCT
- QUANTITY
- TIME
- WORK_OF_ART

Supported Languages

- English
- Chinese
- Danish
- Dutch
- French
- German
- Greek
- Italian
- Japanese
- Lithuanian
- Norwegian Bokmål
- Polish
- Portuguese
- Romanian
- Spanish

Support of custom NER

SpaCy has a step-by-step guide on improving the existing model by adding your data and how to train the model for the new entity type.

Example of API:

```
nlp = spacy.load("en_core_web_lg")
tokenized_text = nlp(doc)
classified_doc = nlp(doc)
```

Other notes

'O' - outside class automatically filtered. The beginning and end position of a named entity is available.

¹<https://en.wikipedia.org/wiki/SpaCy>

B. Stanford NER

Stanford NER² is a Java implementation of a Named Entity Recognizer. Named Entity Recognition (NER) labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. It comes with well-engineered feature extractors for Named Entity Recognition, and many options for defining feature extractors. Included with the download are good named entity recognizers for English, particularly for the 3 classes (PERSON, ORGANIZATION, LOCATION), and we also make available on this page various other models for different languages and circumstances, including models trained on just the CoNLL 2003 English training data.

Stanford NER is also known as CRFClassifier. The software provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, by training your own models on labeled data, you can actually use this code to build sequence models for NER or any other task. (CRF models were pioneered by Lafferty, McCallum, and Pereira (2001); see Sutton and McCallum (2006) or Sutton and McCallum (2010) for more comprehensible introductions.)

Supported NER categories

- Location
- Person
- Organization
- Money
- Percent
- Date
- Time

Supported Languages

- English
- Arabic
- Chinese
- French
- German
- Spanish

Support of custom NER

The documentation for training your classifier is somewhere between bad and non-existent. Nevertheless, everything you need is in the box, and you should look through the Javadoc for at least the classes CRFClassifier and NERFeatureFactory.

Example of API:

```
st = StanfordNERTagger(  
    'english.all.3class.distsim.crf.ser.gz', 'stanford-ner-4.0.0.jar',  
    encoding='utf-8')  
tokenized_doc = word_tokenize(doc)  
classified_doc = st.tag(tokenized_doc)
```

Other notes

The result is a list of tuples for each token without scores. There is no out of the box chunking.

²<https://nlp.stanford.edu/software/CRF-NER.html>

C. Google Cloud Natural Language API

The Cloud Natural Language API³ provides natural language understanding technologies to developers, including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This API is part of the larger Cloud Machine Learning API family.

Supported NER categories

- UNKNOWN
- PERSON
- LOCATION
- ORGANIZATION
- EVENT
- WORK_OF_ART
- CONSUMER_GOOD
- OTHER
- PHONE_NUMBER
- Phone
- ADDRESS
- DATE
- NUMBER
- PRICE

Supported Languages

- English
- Chinese (Simplified)
- Chinese (Traditional)
- French
- German
- Italian
- Japanese
- Korean
- Portuguese (Brazilian & Continental)
- Russian
- Spanish

Support of custom NER

Google has AutoML Natural Language, which allows the end-user to train the custom NER model.

Example of API:

```
client = language.LanguageServiceClient  
    .from_service_account_json('services.json')  
doc = types.Document(content=text,  
    type=language.enums.Document.Type.PLAIN_TEXT)  
features = {'extract_syntax': False,  
    'extract_entities': True,  
    'extract_document_sentiment': False,  
    'extract_entity_sentiment': False,  
    'classify_text': False}  
response = client.annotate_text(doc, features)  
entities = response.entities
```

Other notes

Credentials can be easily exported from the web interface and imported to the client using a JSON file. There is a limitation on the request size.

³<https://cloud.google.com/natural-language/docs>

D. Amazon Comprehend Text Analysis API

Amazon Comprehend⁴ uses natural language processing (NLP) to extract insights about the content of documents. Amazon Comprehend processes any text file in UTF-8 format. It develops insights by recognizing the entities, key phrases, language, sentiments, and other common elements in a document. Use Amazon Comprehend to create new products based on understanding the structure of documents. For example, using Amazon Comprehend you can search social networking feeds for mentions of products or scan an entire document repository for key phrases.

Supported NER categories

- COMMERCIAL_ITEM
- DATE
- EVENT
- LOCATION
- ORGANIZATION
- OTHER
- PERSON
- QUANTITY
- TITLE

Supported Languages

- English
- German
- Spanish
- Italian
- Portuguese
- French
- Japanese
- Korean
- Hindi
- Arabic
- Chinese (simplified)
- Chinese (traditional)

Support of custom NER

Amazon has Comprehend's custom entity recognition service to train the custom NER model. The service automatically tests for the best algorithm and parameters while training the model to use, looking for the most accurate combination of these components.

Example of API:

```
comprehend = boto3.client(
    service_name='comprehend')
comprehend.detect_entities(Text=documents,
    LanguageCode='en')
```

Other notes

Comprehend the client is automatically looking for credentials in the environment variables. There is a 5000 bytes limitation on the request size. API supports both single document and batch modes.

⁴<https://docs.aws.amazon.com/comprehend/latest/dg/what-is.html>

E. Microsoft Azure Text Analytics API

The Text Analytics API⁵ is a cloud-based service that provides Natural Language Processing (NLP) features for text mining and text analysis, including: sentiment analysis, opinion mining, key phrase extraction, language detection, and named entity recognition. The API is a part of Azure Cognitive Services, a collection of machine learning and AI algorithms in the cloud for your development projects. You can use these features with the REST API, or the client library.

Supported NER categories

- Person
- PersonType
- Location
- Organization
- Event
- Product
- Skill
- Address
- PhoneNumber
- Email
- URL
- IP
- DateTime
- Quantity

Supported Languages

- English
- Arabic
- Czech
- Chinese-Simplified
- Chinese-Traditional
- Danish
- Dutch
- Finnish
- French
- German
- Hebrew
- Hungarian
- Italian
- Japanese
- Korean
- Norwegian (Bokmål)
- Polish
- Portuguese (Portugal)
- Portuguese (Brazil)
- Russian
- Spanish
- Swedish
- Turkish

Support of custom NER

Custom NER model training can be done through Azure Language Understanding (LUIS) service. This service is mostly designed for chatbots.

⁵<https://docs.microsoft.com/en-us/azure/cognitive-services/text-analytics/overview>

Example of API:

```
credential = AzureKeyCredential(api_key)
client = TextAnalyticsClient(endpoint,
                             credential)
client.recognize_entities(documents,
                          language="en")
```

Other notes

To run any of the Azure Text Analytics containers, you must have the host computer and container environments. There is a 5120 character limitation on the request size. API supports both single document and batch modes.

II. DATASET AND ANALYSIS METHODOLOGY

This dataset is a document annotation dataset used to perform NER on resumes from indeed.com, which was obtained from <https://www.kaggle.com/dataturks/resume-entities-for-ner/home>.



Fig. 1. Example of a tagged resume.

I used the 'Name' tag as PERSON class, both 'College Name' and 'Companies worked at' as ORGANIZATION class and 'Location' as LOCATION class.

III. RESULTS

Amazon shows the best results for this dataset. Google has the smallest precision. It produces a lot of FP for the PERSON class due to the nature of the pre-build model. It extracts values such as 'individual', 'user', 'player', 'member',

'client', 'librarian', 'subject matter expert', 'engineer' which can be useful in some cases e.g. chatbots. Also, in some cases, titles are additionally extracted with names e.g. 'Abhishek Jha Application Development Associate'. SpaCY and Stanford NER partially extract values or extract extra tokens, leading to additional FP and FN errors, affecting both precision and recall.

TABLE I
RESULTS (PERSON)

NLP Tool	Person					
	TP	FP	FN	Precision	Recall	F1 Score
SpaCy	184	470	34	0.281	0.844	0.422
Stanford	57	268	161	0.175	0.261	0.210
Google	69	2058	149	0.032	0.317	0.059
Amazon	180	168	38	0.517	0.826	0.636
Microsoft	68	122	150	0.358	0.312	0.333

Overall the scores for all NLP toolkits are pretty low due to the nature of the data. Technologies listed in the resume were not labeled as ORGANIZATION however, technically, 'Microsoft', 'Cisco', 'Oracle', 'Apache' etc. are organizations. That's why we see a lot of FP errors. All models struggle to identify value boundaries correctly. Amazon is showing the best results again. Google has the same issues as with PERSON. It extracts values such as 'company', 'school', 'organization'.

TABLE II
RESULTS (ORGANIZATION)

NLP Tool	Organization					
	TP	FP	FN	Precision	Recall	F1 Score
SpaCy	157	6030	375	0.025	0.295	0.047
Stanford	84	2169	448	0.037	0.158	0.060
Google	165	3335	367	0.047	0.310	0.082
Amazon	195	1979	337	0.090	0.367	0.144
Microsoft	27	772	505	0.034	0.051	0.041

Manual analysis of the model errors showed good results. Most of the FP errors were actual locations, but they were not labeled for this task. SpaCy showed the best results for the LOCATION class for this dataset. A lot of FP errors Google again extracts values such as 'region', 'center', 'site', 'floor', 'area', 'shop'. Only opensource models produced errors such as labeling 'Java' as a location.

TABLE III
RESULTS (LOCATION)

NLP Tool	Location					
	TP	FP	FN	Precision	Recall	F1 Score
SpaCy	143	795	60	0.152	0.704	0.251
Stanford	27	851	176	0.031	0.133	0.050
Google	133	1478	70	0.083	0.655	0.147
Amazon	44	978	159	0.043	0.217	0.072
Microsoft	98	737	105	0.117	0.483	0.189

You can find the source code and results here https://github.com/ArtsiomStrok/tech_review.