

contentSkip to site indexPoliticsSubscribeLog InSubscribeLog InToday's PaperAdvertisementSupported ORG byF.B.I. Agent Peter Strzok PERSON , Who Criticized Trump PERSON in Texts, Is FiredImagePeter Strzok, a top F.B.I. GPE counterintelligence agent who was taken off the special counsel investigation after his disparaging texts about President Trump PERSON were uncovered, was fired. CreditT.J. Kirkpatrick PERSON for The New York TimesBy Adam Goldman ORG and Michael S. SchmidtAug PERSON . 13 CARDINAL , 2018WASHINGTON CARDINAL — Peter Strzok PERSON , the F.B.I. GPE senior counterintelligence agent who disparaged President Trump PERSON in inflammatory text messages and helped oversee the Hillary Clinton PERSON email and Russia GPE investigations, has been fired for violating bureau policies, Mr. Strzok PERSON 's lawyer said Monday DATE .Mr. Trump and his allies seized on the texts — exchanged during the 2016 DATE campaign with a former F.B.I. GPE lawyer, Lisa Page — in PERSON assailing the Russia GPE investigation as an illegitimate "witch hunt." Mr. Strzok PERSON , who rose over 20 years DATE at the F.B.I. GPE to become one of its most experienced counterintelligence agents, was a key figure in the early months DATE of the inquiry.Along with writing the texts, Mr. Strzok PERSON was accused of sending a highly sensitive search warrant to his personal email account.The F.B.I. GPE had been under immense political pressure by Mr. Trump PERSON to dismiss Mr. Strzok PERSON , who was removed last summer



Search Medium



Named Entity Recognition and Classification with Scikit-Learn

How to train machine learning models for NER using Scikit-Learn's libraries



Susan Li · Following

Published in Towards Data Science · 7 min read · Aug 27, 2018



1.94K



11



...

Named Entity Recognition and Classification (NERC) is a process of recognizing information units like names, including person, organization and location names, and numeric expressions including time, date, money and percent expressions from unstructured text. The goal is to develop practical and domain-independent techniques in order to detect named entities with high accuracy automatically.

Last week, we gave [an introduction on Named Entity Recognition \(NER\) in NLTK and SpaCy](#). Today, we go a step further, — training machine learning models for NER using some of Scikit-Learn's libraries. Let's get started!

The Data

The data is feature engineered corpus annotated with [IOP](#) and [POS](#) tags that can be found at [Kaggle](#). We can have a quick peek of first several rows of the data.

A Sentence #	A Word	A POS	A Tag
Sentence: 1	Thousands	NNS	0
	of	IN	0
	demonstrators	NNS	0
	have	VBP	0
	marched	VBN	0
	through	IN	0
	London	NNP	B-geo
	to	TO	0
	protest	VB	0
	the	DT	0
	war	NN	0
	in	IN	0
	Iraq	NNP	B-geo
	and	CC	0
	demand	VB	0
	the	DT	0
	withdrawal	NN	0
	of	IN	0
	British	JJ	B-gpe
	troops	NNS	0
	from	IN	0
	that	DT	0
	country	NN	0
	.	.	0
Sentence: 2	Families	NNS	0
	of	IN	0
	soldiers	NNS	0

Figure 1

Essential info about entities:

- geo = Geographical Entity
- org = Organization
- per = Person
- gpe = Geopolitical Entity

- tim = Time indicator
- art = Artifact
- eve = Event
- nat = Natural Phenomenon

Inside–outside–beginning (tagging)

The IOB (short for inside, outside, beginning) is a common tagging format for tagging tokens.

- I- prefix before a tag indicates that the tag is inside a chunk.
- B- prefix before a tag indicates that the tag is the beginning of a chunk.
- An O tag indicates that a token belongs to no chunk (outside).

```
import pandas as pd
import numpy as np
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.linear_model import Perceptron
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
```

The entire data set can not be fit into the memory of a single computer, so we select the first 100,000 records, and use Out-of-core learning algorithms to efficiently fetch and process the data.

```
df = pd.read_csv('ner_dataset.csv', encoding = "ISO-8859-1")
df = df[:100000]
df.head()
```

	Sentence #	Word	POS	Tag
0	Sentence: 1	Thousands	NNS	O
1	NaN	of	IN	O
2	NaN	demonstrators	NNS	O
3	NaN	have	VBP	O
4	NaN	marched	VBN	O

Figure 2

```
df.isnull().sum()
```

Sentence #	95456
Word	0
POS	0
Tag	0
dtype:	int64

Figure 3

Data Preprocessing

We notice that there are many NaN values in ‘Sentence #’ column, and we fill NaN by preceding values.

```
df = df.fillna(method='ffill')  
df['Sentence #'].nunique(), df.Word.nunique(), df.Tag.nunique()
```

(4544, 10922, 17)

We have 4,544 sentences that contain 10,922 unique words and tagged by 17 tags.

The tags are not evenly distributed.

```
df.groupby('Tag').size().reset_index(name='counts')
```

Tag counts		
0	B-art	75
1	B-eve	53
2	B-geo	3303
3	B-gpe	1740
4	B-nat	30
5	B-org	1876
6	B-per	1668
7	B-tim	1823
8	I-art	43
9	I-eve	47
10	I-geo	690
11	I-gpe	51
12	I-nat	11
13	I-org	1470
14	I-per	1846
15	I-tim	549
16	O	84725

Figure 4

The following code transform the text date to vector using DictVectorizer and then split to train and test sets.

```
X = df.drop('Tag', axis=1)
v = DictVectorizer(sparse=False)
```

```
X = v.fit_transform(X.to_dict('records'))
y = df.Tag.values

classes = np.unique(y)
classes = classes.tolist()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.33, random_state=0)
X_train.shape, y_train.shape
```

((67000, 15507), (67000,))

Out-of-core Algorithms

We will try some of the out-of-core algorithms that are designed to process data that is too large to fit into a single computer memory that support `partial_fit` method.

Perceptron

```
per = Perceptron(verbose=10, n_jobs=-1, max_iter=5)
per.partial_fit(X_train, y_train, classes)
```

```
Total training time: 2.52 seconds.

Total training time: 2.50 seconds.

Total training time: 2.49 seconds.
Norm: 13.53, NNZs: 159, Bias: -3.000000, T: 67000, Avg. loss: 0.001701
Total training time: 2.53 seconds.
-- Epoch 1-- Epoch 1
-- Epoch 1

-- Epoch 1
Norm: 8.37, NNZs: 59, Bias: -2.000000, T: 67000, Avg. loss: 0.000522
Total training time: 1.77 seconds.
-- Epoch 1
Norm: 45.00, NNZs: 1164, Bias: -3.000000, T: 67000, Avg. loss: 0.017567
Total training time: 1.79 seconds.
-- Epoch 1
Norm: 48.33, NNZs: 1679, Bias: -4.000000, T: 67000, Avg. loss: 0.022507
Total training time: 1.79 seconds.
-- Epoch 1
Norm: 57.04, NNZs: 2028, Bias: -5.000000, T: 67000, Avg. loss: 0.034493
Total training time: 1.85 seconds.
-- Epoch 1

[Parallel(n_jobs=-1)]: Done  5 tasks      | elapsed:   4.5s
Norm: 10.15, NNZs: 83, Bias: -3.000000, T: 67000, Avg. loss: 0.000806
Total training time: 1.68 seconds.
-- Epoch 1
Norm: 10.30, NNZs: 92, Bias: -2.000000, T: 67000, Avg. loss: 0.001030
Total training time: 1.69 seconds.
-- Epoch 1
Norm: 34.35, NNZs: 811, Bias: -4.000000, T: 67000, Avg. loss: 0.011851
Total training time: 1.71 seconds.
-- Epoch 1
Norm: 10.72, NNZs: 93, Bias: -3.000000, T: 67000, Avg. loss: 0.001194
Total training time: 1.68 seconds.
-- Epoch 1

[Parallel(n_jobs=-1)]: Done  10 tasks     | elapsed:   6.2s
[Parallel(n_jobs=-1)]: Done  12 out of  17 | elapsed:   6.2s remaining:   2.5s
Norm: 6.40, NNZs: 33, Bias: -3.000000, T: 67000, Avg. loss: 0.000194
Total training time: 1.94 seconds.
-- Epoch 1
Norm: 52.49, NNZs: 1692, Bias: -4.000000, T: 67000, Avg. loss: 0.025776
Total training time: 2.02 seconds.
Norm: 31.94, NNZs: 698, Bias: -4.000000, T: 67000, Avg. loss: 0.011791
Total training time: 2.01 seconds.
Norm: 60.29, NNZs: 2085, Bias: -5.000000, T: 67000, Avg. loss: 0.026746
Total training time: 2.09 seconds.

[Parallel(n_jobs=-1)]: Done  14 out of  17 | elapsed:   8.2s remaining:   1.7s
Norm: 73.60, NNZs: 2820, Bias: 3.000000, T: 67000, Avg. loss: 0.048672
Total training time: 1.48 seconds.

[Parallel(n_jobs=-1)]: Done  17 out of  17 | elapsed:   9.6s finished

Out[16]: Perceptron(alpha=0.0001, class_weight=None, eta0=1.0, fit_intercept=True,
max_iter=5, n_iter=None, n_jobs=-1, penalty=None, random_state=0,
shuffle=True, tol=None, verbose=10, warm_start=False)
```

Figure 5

Because tag “O” (outside) is the most common tag and it will make our results look much better than they actual are. So we remove tag “O” when we evaluate classification metrics.

```
new_classes = classes.copy()
new_classes.pop()
new_classes
```

```
['B-art',
 'B-eve',
 'B-geo',
 'B-gpe',
 'B-nat',
 'B-org',
 'B-per',
 'B-tim',
 'I-art',
 'I-eve',
 'I-geo',
 'I-gpe',
 'I-nat',
 'I-org',
 'I-per',
 'I-tim']
```

Figure 6

```
print(classification_report(y_pred=per.predict(X_test),
y_true=y_test, labels=new_classes))
```

	precision	recall	f1-score	support
B-art	0.15	0.12	0.14	24
B-eve	0.46	0.32	0.37	19
B-geo	0.42	0.91	0.57	1085
B-gpe	0.89	0.78	0.83	556
B-nat	0.11	0.25	0.15	12
B-org	0.55	0.35	0.43	589
B-per	0.72	0.43	0.53	564
B-tim	0.65	0.78	0.71	611
I-art	0.02	0.08	0.03	12
I-eve	0.00	0.00	0.00	18
I-geo	0.81	0.32	0.46	230
I-gpe	0.00	0.00	0.00	14
I-nat	0.50	0.50	0.50	2
I-org	0.71	0.41	0.52	445
I-per	0.76	0.20	0.32	591
I-tim	0.26	0.05	0.09	194
avg / total	0.62	0.55	0.53	4966

Figure 7

Linear classifiers with SGD training

```

sgd = SGDClassifier()
sgd.partial_fit(X_train, y_train, classes)

SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
    eta0=0.0, fit_intercept=True, l1_ratio=0.15,
    learning_rate='optimal', loss='hinge', max_iter=None, n_iter=None,
    n_jobs=1, penalty='l2', power_t=0.5, random_state=None,
    shuffle=True, tol=None, verbose=0, warm_start=False)

```

Figure 8

```
print(classification_report(y_pred=sgd.predict(X_test),
y_true=y_test, labels=new_classes))
```

	precision	recall	f1-score	support
B-art	0.33	0.12	0.18	24
B-eve	0.67	0.11	0.18	19
B-geo	0.76	0.66	0.71	1085
B-gpe	0.86	0.63	0.73	556
B-nat	0.67	0.33	0.44	12
B-org	0.63	0.42	0.50	589
B-per	0.61	0.55	0.58	564
B-tim	0.79	0.62	0.70	611
I-art	1.00	0.08	0.15	12
I-eve	0.00	0.00	0.00	18
I-geo	0.82	0.39	0.53	230
I-gpe	0.50	0.07	0.12	14
I-nat	0.00	0.00	0.00	2
I-org	0.36	0.68	0.47	445
I-per	0.59	0.67	0.63	591
I-tim	1.00	0.01	0.02	194
avg / total	0.69	0.56	0.59	4966

Figure 9

Naive Bayes classifier for multinomial models

```
nb = MultinomialNB(alpha=0.01)
nb.partial_fit(X_train, y_train, classes)
```

```
MultinomialNB(alpha=0.01, class_prior=None, fit_prior=True)
```

Figure 10

```
print(classification_report(y_pred=nb.predict(X_test), y_true=y_test,
labels = new_classes))
```

	precision	recall	f1-score	support
B-art	0.06	0.17	0.09	24
B-eve	0.33	0.37	0.35	19
B-geo	0.70	0.63	0.66	1085
B-gpe	0.70	0.83	0.76	556
B-nat	0.35	0.50	0.41	12
B-org	0.41	0.44	0.43	589
B-per	0.44	0.47	0.46	564
B-tim	0.56	0.61	0.59	611
I-art	0.07	0.08	0.08	12
I-eve	0.46	0.33	0.39	18
I-geo	0.40	0.52	0.46	230
I-gpe	0.13	0.14	0.14	14
I-nat	0.00	0.00	0.00	2
I-org	0.50	0.51	0.51	445
I-per	0.53	0.50	0.51	591
I-tim	0.17	0.27	0.21	194
avg / total	0.54	0.56	0.54	4966

Figure 11

Passive Aggressive Classifier

```
pa =PassiveAggressiveClassifier()
pa.partial_fit(X_train, y_train, classes)
```

```
PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
    fit_intercept=True, loss='hinge', max_iter=None, n_iter=None,
    n_jobs=1, random_state=None, shuffle=True, tol=None,
    verbose=0, warm_start=False)
```

Figure 12

```
print(classification_report(y_pred=pa.predict(X_test), y_true=y_test,
labels=new_classes))
```

	precision	recall	f1-score	support
B-art	0.20	0.04	0.07	24
B-eve	0.36	0.26	0.30	19
B-geo	0.70	0.65	0.67	1085
B-gpe	0.60	0.85	0.70	556
B-nat	0.20	0.67	0.31	12
B-org	0.70	0.32	0.44	589
B-per	0.57	0.56	0.57	564
B-tim	0.84	0.62	0.71	611
I-art	0.02	0.50	0.04	12
I-eve	0.83	0.28	0.42	18
I-geo	0.50	0.62	0.55	230
I-gpe	0.40	0.43	0.41	14
I-nat	0.20	0.50	0.29	2
I-org	0.78	0.28	0.41	445
I-per	0.63	0.64	0.63	591
I-tim	0.21	0.32	0.25	194
avg / total	0.65	0.56	0.58	4966

Figure 13

None of the above classifiers produced satisfying results. It is obvious that it is not going to be easy to classify named entities using regular classifiers.

Conditional Random Fields (CRFs)

CRFs is often used for labeling or parsing of sequential data, such as natural language processing and CRFs find applications in POS Tagging, named entity recognition, among others.

sklearn-crfsuite

We will train a CRF model for named entity recognition using sklearn-crfsuite on our data set.

```
import sklearn_crfsuite
from sklearn_crfsuite import scorers
```

```
from sklearn_crfsuite import metrics
from collections import Counter
```

The following code is to retrieve sentences with their POS and tags. Thanks [Tobias](#) for the tip.

```
class SentenceGetter(object):

    def __init__(self, data):
        self.n_sent = 1
        self.data = data
        self.empty = False
        agg_func = lambda s: [(w, p, t) for w, p, t in
            zip(s['Word'].values.tolist(),
                s['POS'].values.tolist(),
                s['Tag'].values.tolist())]
        self.grouped = self.data.groupby('Sentence #').apply(agg_func)
        self.sentences = [s for s in self.grouped]

    def get_next(self):
        try:
            s = self.grouped['Sentence: {}'.format(self.n_sent)]
            self.n_sent += 1
            return s
        except:
            return None

getter = SentenceGetter(df)
sentences = getter.sentences
```

Feature Extraction

Next, we extract more features (word parts, simplified POS tags, lower/title/upper flags, features of nearby words) and convert them to **sklearn-crfsuite** format — each sentence should be converted to a list of dicts. The following code were taken from [sklearn-crfsuitess official site](#).

```

def word2features(sent, i):
    word = sent[i][0]
    postag = sent[i][1]

    features = {
        'bias': 1.0,
        'word.lower()': word.lower(),
        'word[-3:)': word[-3:],
        'word[-2:)': word[-2:],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'postag': postag,
        'postag[:2]': postag[:2],
    }
    if i > 0:
        word1 = sent[i-1][0]
        postag1 = sent[i-1][1]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            '-1:postag': postag1,
            '-1:postag[:2]': postag1[:2],
        })
    else:
        features['BOS'] = True
    if i < len(sent)-1:
        word1 = sent[i+1][0]
        postag1 = sent[i+1][1]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            '+1:postag': postag1,
            '+1:postag[:2]': postag1[:2],
        })
    else:
        features['EOS'] = True

    return features

def sent2features(sent):
    return [word2features(sent, i) for i in range(len(sent))]

def sent2labels(sent):
    return [label for token, postag, label in sent]

def sent2tokens(sent):
    return [token for token, postag, label in sent]

```

Split train and test sets

```
X = [sent2features(s) for s in sentences]
y = [sent2labels(s) for s in sentences]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.33, random_state=0)
```

Train a CRF model

```
crf = sklearn_crfsuite.CRF(
    algorithm='lbgf',
    c1=0.1,
    c2=0.1,
    max_iterations=100,
    all_possible_transitions=True
)
crf.fit(X_train, y_train)

CRF(algorithm='lbgf', all_possible_states=None,
    all_possible_transitions=True, averaging=None, c=None, c1=0.1, c2=0.1,
    calibration_candidates=None, calibration_eta=None,
    calibration_max_trials=None, calibration_rate=None,
    calibration_samples=None, delta=None, epsilon=None, error_sensitive=None,
    gamma=None, keep_tempfiles=None, linesearch=None, max_iterations=100,
    max_linesearch=None, min_freq=None, model_filename=None,
    num_memories=None, pa_type=None, period=None, trainer_cls=None,
    variance=None, verbose=False)
```

Figure 14

Evaluation

```
y_pred = crf.predict(X_test)
print(metrics.flat_classification_report(y_test, y_pred, labels =
new_classes))
```

	precision	recall	f1-score	support
B-art	1.00	0.03	0.07	29
B-eve	0.86	0.25	0.39	24
B-geo	0.75	0.88	0.81	1043
B-gpe	0.89	0.78	0.83	588
B-nat	0.67	0.20	0.31	10
B-org	0.75	0.64	0.69	649
B-per	0.81	0.81	0.81	546
B-tim	0.90	0.85	0.87	589
I-art	0.00	0.00	0.00	7
I-eve	0.57	0.22	0.32	18
I-geo	0.71	0.71	0.71	204
I-gpe	0.47	0.53	0.50	17
I-nat	1.00	0.50	0.67	2
I-org	0.78	0.73	0.76	545
I-per	0.80	0.90	0.85	574
I-tim	0.79	0.68	0.73	185
avg / total	0.80	0.78	0.78	5030

Figure 15

Way better! We will stick to sklearn-crfsuite and explore more!

What our classifier learned?

```
def print_transitions(trans_features):
    for (label_from, label_to), weight in trans_features:
        print("%-6s -> %-7s %0.6f" % (label_from, label_to, weight))

print("Top likely transitions:")
print_transitions(Counter(crf.transition_features_).most_common(20))

print("\nTop unlikely transitions:")
print_transitions(Counter(crf.transition_features_).most_common()
[-20:])
```

Top likely transitions:

B-geo	-> I-geo	6.007604
I-geo	-> I-geo	5.296245
B-art	-> I-art	4.951198
B-eve	-> I-eve	4.847021
I-tim	-> I-tim	4.789188
B-per	-> I-per	4.711716
I-art	-> I-art	4.664539
B-tim	-> I-tim	4.575079
B-org	-> I-org	4.456466
I-org	-> I-org	4.320635
I-per	-> I-per	4.039724
I-gpe	-> I-gpe	3.969627
I-eve	-> I-eve	3.968368
B-gpe	-> I-gpe	3.919860
O	-> O	3.465068
B-nat	-> I-nat	3.208265
O	-> B-per	2.057576
B-org	-> B-art	2.001540
I-nat	-> I-nat	1.919624
B-geo	-> B-tim	1.688412

Top unlikely transitions:

B-gpe	-> I-org	-1.848015
O	-> I-gpe	-1.856660
B-geo	-> I-gpe	-1.880598
I-per	-> I-org	-1.889957
B-geo	-> I-org	-1.947059
O	-> I-eve	-2.033728
B-gpe	-> I-geo	-2.151673
I-org	-> B-org	-2.177301
B-org	-> B-org	-2.258343
O	-> I-art	-2.325744
B-org	-> I-per	-2.332204
B-tim	-> B-tim	-2.447829
I-org	-> I-per	-2.455738
I-per	-> B-per	-3.094530
O	-> I-per	-3.122940
B-gpe	-> B-gpe	-3.169217
O	-> I-tim	-4.152981
O	-> I-geo	-4.235485
B-per	-> B-per	-4.278895
O	-> I-org	-4.543933

Figure 16

Interpretation: It is very likely that the beginning of a geographical entity (B-geo) will be followed by a token inside geographical entity (I-geo), but transitions to inside of an organization name (I-org) from tokens with other labels are penalized hugely.

Check the state features

```
def print_state_features(state_features):
    for (attr, label), weight in state_features:
        print("%0.6f %-8s %s" % (weight, label, attr))

print("Top positive:")
print_state_features(Counter(crf.state_features_).most_common(30))

print("\nTop negative:")
print_state_features(Counter(crf.state_features_).most_common()
[-30:])
```

Top positive:

5.183603	B-tim	word[-3:]:day
4.699027	O	BOS
3.761687	O	bias
3.754395	I-tim	word[-3:]:day
3.593121	O	word.lower():kurdish
3.584948	O	word.lower():jewish
3.370614	B-per	word.lower():president
3.338913	B-org	word.lower():al-qaida
3.326234	B-tim	word.lower():thanksgiving
3.269326	B-tim	word[-2:]:ay
3.225759	O	word[-2:]:N1
3.171786	B-tim	+1:word.lower():year
3.119587	B-tim	word.lower():afternoon
3.118231	O	postag[:2]:VB
3.090609	B-org	-1:word.lower():telephoned
3.081398	B-org	word.lower():hamas
3.050390	B-gpe	word.istitle()
3.037740	B-tim	word[-2:]:0s
3.023566	B-gpe	word.lower():nepal
3.003322	B-gpe	word[-3:]:pal
2.998838	B-org	+1:word.lower():fought
2.997746	I-geo	+1:word.lower():town
2.995321	B-per	word.lower():obama
2.980474	B-geo	word.lower():mid-september
2.929354	B-geo	-1:word.lower():serb
2.924037	I-geo	+1:word.lower():achieved
2.921243	B-per	BOS
2.915479	B-tim	+1:word.lower():czech
2.898837	B-org	-1:word.lower():brunei
2.888846	O	word.lower():last

Top negative:

-1.985977 0	+1:word.lower():hours
-2.019767 0	+1:word.lower():moscow
-2.034469 0	-1:word.lower():year
-2.041179 0	word.lower():32-year-old
-2.058087 0	word.lower():later
-2.105580 0	+1:word.lower():weeks
-2.155248 0	+1:word.lower():monday
-2.156750 0	word[-3:]:oon
-2.180948 0	word.lower():evening
-2.182285 0	word.lower():another
-2.191040 0	word.isupper()
-2.197670 0	word.lower():prime
-2.269735 0	-1:word.lower():doubled
-2.297358 0	word.lower():decade
-2.349126 0	-1:word.lower():brunei
-2.349738 0	word.lower():anniversary
-2.418581 0	+1:word.lower():influence
-2.427463 0	-1:word.lower():extremist
-2.431002 0	+1:word.lower():czech
-2.526648 B-geo	-1:word.lower():recognize
-2.645574 0	+1:word.lower():mr.
-2.647633 0	+1:word.lower():months
-2.708926 0	word.lower():morning
-2.766577 0	+1:word.lower():years
-2.818992 0	+1:word.lower():year
-3.025051 0	+1:word.lower():last
-3.087828 0	word.isdigit()
-3.233526 0	word.istitle()
-3.521244 0	postag>NNP
-3.895403 0	word[-2:]:0s

Figure 17

Observations:

1). **5.183603 B-tim word[-3]:day** The model learns that if a nearby word was “day” then the token is likely a part of a Time indicator.

2). **3.370614 B-per word.lower():president** The model learns that token “president” is likely to be at the beginning of a person name.

3). -3.521244 0 postag:NNP The model learns that proper nouns are often entities.

4). -3.087828 0 word.isdigit() Digits are likely entities.

5). -3.233526 0 word.istitle() TitleCased words are likely entities.

ELI5

ELI5 is a Python package which allows to check weights of sklearn_crfsuite.CRF models.

Inspect model weights

```
import eli5
eli5.show_weights(crf, top=10)
```

From \ To	O	B-art	I-art	B-eve	I-eve	B-geo	I-geo	B-gpe	I-gpe	B-nat	I-nat	B-org	I-org	B-per	I-per	B-tim	I-tim
O	3.465	0.477	-2.326	0.973	-2.034	0.919	-4.235	0.506	-1.857	0.049	-1.256	0.794	-4.544	2.058	-3.123	1.417	-4.153
B-art	-0.876	-0.023	4.951	-0.003	-0.101	-0.373	-0.232	-0.373	-0.251	-0.008	-0.08	0.606	-0.601	-0.816	-0.784	-0.669	-0.324
I-art	-0.986	-0.279	4.665	-0.014	-0.086	0.336	-0.262	-0.272	-0.089	-0.008	-0.066	-0.44	-0.52	-0.747	-0.563	0.093	-0.399
B-eve	-0.533	-0.006	-0.077	-0.022	4.847	-0.234	-0.219	-0.328	-0.177	0.0	-0.04	-0.479	-0.504	-0.844	-0.409	-0.656	-0.515
I-eve	-0.333	0.0	-0.034	-0.653	3.968	-0.257	-0.193	-0.105	-0.059	-0.01	-0.009	-0.233	-0.272	-0.351	-0.387	-0.384	-0.177
B-geo	0.216	1.413	-1.024	-0.136	-0.695	-1.541	6.008	1.1	-1.881	-0.05	-0.502	-1.03	-1.947	-0.966	-1.813	1.688	-1.373
I-geo	-0.034	-0.048	-0.417	-0.029	-0.256	-1.011	5.296	-0.468	-0.719	-0.009	-0.147	-0.786	-1.018	-0.791	-0.642	1.238	-0.928
B-gpe	0.62	-0.255	-0.858	-0.278	-0.661	-0.184	-2.152	-3.169	3.92	-0.049	-0.296	0.951	-1.848	0.572	-1.357	-0.347	-0.987
I-gpe	-0.656	-0.163	-0.082	-0.01	-0.031	-0.007	-0.61	-0.624	3.97	0.0	-0.024	-0.377	-0.622	-0.619	-0.441	-0.684	-0.247
B-nat	-0.405	-0.001	-0.055	0.0	-0.042	-0.254	-0.109	-0.182	-0.068	-0.005	3.208	-0.255	-0.334	-0.55	-0.394	-0.231	-0.078
I-nat	-0.835	-0.002	-0.037	0.0	-0.007	-0.18	-0.053	-0.093	-0.026	-0.066	1.92	-0.133	-0.227	-0.364	-0.231	-0.182	-0.04
B-org	0.046	2.002	-1.136	-0.195	-0.816	-0.611	-1.839	-0.26	-1.572	-0.129	-0.703	-2.258	4.456	-0.771	-2.332	-0.652	-1.306
I-org	0.042	-0.319	-0.961	-0.174	-0.68	-1.657	-1.318	-0.708	-0.912	-0.434	-0.591	-2.177	4.321	-0.133	-2.456	0.119	-1.327
B-per	0.016	-0.302	-0.773	-0.174	-0.758	0.028	-1.0	0.617	-1.042	-0.095	-0.668	0.918	-1.698	-4.279	4.712	-0.386	-0.846
I-per	-0.223	-0.169	-0.683	-0.278	-0.747	-1.268	-1.189	-0.71	-0.974	-0.078	-0.593	-1.132	-1.89	-3.095	4.04	0.177	-1.16
B-tim	0.311	-0.451	-0.4	-0.059	-0.557	-0.759	-0.991	-1.165	-0.447	0.611	-0.252	-0.629	-1.229	-1.309	-0.897	-2.448	4.575
I-tim	0.145	-0.144	-0.142	-0.356	-0.15	0.62	-0.291	0.037	-0.067	-0.064	-0.017	-0.812	-0.74	-0.006	-0.224	-1.571	4.789

y=O top features		y=B-art top features		y=I-art top features		y=B-eve top features		y=I-eve top features		y=B-geo top features		y=I-geo top fe	
Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feature	Weight?	Feat
+4.699	BOS	+2.250	word.lower	+1.221	word.lower	+2.018	-1:word.lower	+1.133	word.lower	+2.980	word.lower	+2.998	+1:w
+3.762	bias	+2.203	(:)twitter	+1.220	(:)constitution	+1.604	-1:word.lower	+1.132	(:)war	+2.924	(:)mid-	+2.924	(:)to
+3.593	word.lower	+2.203	word.lower	+1.220	(:)english	+1.604	(:)airport	+1.132	(:)postag:NNPS	+2.924	september	+2.924	+1:w
+3.585	(:)kurdish	+2.038	-1:word.lower	+1.076	(:)tamilnet	+1.530	-1:word.lower	+1.122	(:)first	+2.929	-1:word.lower	+2.472	(:)ac
+3.226	word[-2]:N1	+1.649	(:)jewish	+1.075	word.lower	+1.523	(:)magazine	+1.111	(:)celebrated	+2.882	word.lower	+2.221	+1:w
+3.118	postag[:2]:VB	+1.649	(:)tamilnet	+1.075	word.lower	+1.523	word[-2]:Us	+1.030	(:)christmas	+2.623	(:)serb	+2.472	(:)blk
... 6280 more positive 1789 more negative 17 more negative 12 more negative 13 more negative 275 more negative 726 more ne	
... 22 more negative 109 more ne	
-3.088	word.isdigit()	+1.562	-1:word.lower	+1.028	(:)newspaper	+1.429	(:)games	+0.991	(:)pen	+2.616	(:)china	+2.113	(:)w
-3.234	word.istitle()	+1.536	-1:word.lower	+0.988	(:)unlike	+1.366	(:)christmas	+0.988	(:)open	+2.531	(:)aswat	+2.221	(:)se
-3.521	postag>NNP	+1.536	(:)would	+0.961	(:)simple	+1.347	(:)mas	+0.979	(:)rs	+2.503	(:)washington	+1.879	-1:w
-3.895	word[-2]:Os	+1.528	word[-3]:eep	+1.347	(:)zahedian	+1.347	word.lower():ii	+1.347	(:)zahedian	+2.451	(:)beijing	+1.805	+1:w
... 212 more positive 192 more positive 106 more positive 108 more positive 1699 more positive 17 more negative 17 more negative ...	
... 22 more negative 12 more negative 13 more negative 13 more negative 275 more negative 726 more ne		... 109 more ne	

Figure 18

Observations:

- It does make sense that I-entity must follow B-entity, such as I-geo follows B-geo, I-org follows B-org, I-per follows B-per, and so on.
- We can also see that it is not common in this data set to have a person right after an organization name (B-org -> I-per has a large negative weight).
- The model learned large negative weights for impossible transitions like O -> I-geo, O -> I-org and O -> I-tim, and so on.

For easy to read, we can check only a subset of tags.

```
eli5.show_weights(crf, top=10, targets=['O', 'B-org', 'I-per'])
```

From \ To **O** **B-org** **I-per**

	O	0.561	0.795	-3.017
B-org	0.086	-2.285	-2.309	
I-per	-0.066	-1.147	3.898	

y=O top features		y=B-org top features		y=I-per top features	
Weight?	Feature	Weight?	Feature	Weight?	Feature
+4.720	BOS	+3.603	word.lower():al-qaida	+1.788	+1:word.lower():david
+4.236	bias	+3.340	word.lower():hamas	+1.709	+1:word.lower():saad
+4.046	word.lower():jewish	+3.270	word.lower():parliament	+1.659	+1:word.lower():reports
+3.494	word.lower():kurdish	+3.041	-1:word.lower():telephoned	+1.519	+1:word.lower():clinton
+3.435	word[-2:]N1	+2.991	-1:word.lower():brunei	+1.491	-1:postag:NN
+3.031	+1:word.lower():minister	+2.966	+1:word.lower():fought	+1.482	word.lower():rice
... 1685 more positive ...		+2.894	word[-3:]ban	+1.385	-1:word.lower():masjid
... 983 more negative ...		+2.754	-1:word.lower():extremist	... 858 more positive ...	
-3.080	+1:word.lower():last	+2.654	+1:word.lower():influence	... 182 more negative ...	
-3.387	word.istitle()	+2.630	word.lower():westerners	-1.365	bias
-4.194	postag>NNP	... 1243 more positive ...		-1.463	+1:postag:NN
-4.325	word[-2:]Os	... 169 more negative ...		-1.588	word[-3:]ion

Figure 19

Or check only some of the features for all tags.

```
eli5.show_weights(crf, top=10, feature_re='^word\\.is',
                  horizontal_layout=False, show=['targets'])
```

y=O top features

Weight?	Feature
-2.186	word.isupper()
-2.720	word.isdigit()
-3.387	word.istitle()

y=B-art top features

Weight?	Feature
+0.151	word.istitle()
-0.214	word.isupper()

y=l-art top features

Weight?	Feature
+0.607	word.istitle()
+0.597	word.isdigit()

y=B-eve top features

Weight?	Feature
+1.185	word.isupper()
+0.391	word.isdigit()
-0.200	word.istitle()

y=l-eve top features

Weight?	Feature
+0.919	word.isupper()
+0.069	word.istitle()

y=B-geo top features

Weight?	Feature
+1.264	word.istitle()
-0.046	word.isupper()
-0.728	word.isdigit()

y=l-geo top features

Weight?	Feature
+0.726	word.istitle()
+0.534	word.isdigit()
-0.000	word.isupper()

y=B-gpe top features

Weight?	Feature
+2.970	word.istitle()
+1.333	word.isupper()

y=l-gpe top features

Weight?	Feature
+0.210	word.istitle()
-0.167	word.isupper()

y=B-nat top features

Weight?	Feature
+1.622	word.isupper()
-0.252	word.istitle()

y=l-nat top features

Weight?	Feature
+0.003	word.istitle()

y=B-org top features

Weight?	Feature
+1.978	word.isupper()
+0.000	word.istitle()
-0.804	word.isdigit()

y=l-org top features

Weight?	Feature
+0.366	word.istitle()
+0.021	word.isupper()
-0.443	word.isdigit()

y=B-per top features

Weight?	Feature
+0.146	word.istitle()
-0.098	word.isdigit()
-1.003	word.isupper()

y=I-per top features

Weight?	Feature
+0.208	word.istitle()
-0.020	word.isdigit()
-0.391	word.isupper()

y=B-tim top features

Weight?	Feature
+2.573	word.isdigit()
-0.435	word.istitle()
-1.133	word.isupper()

y=I-tim top features

Weight?	Feature
+1.978	word.isdigit()
-0.286	word.isupper()
-1.304	word.istitle()

Figure 20

That was it, for now. I enjoyed making my hands dirty on sklearn-crfsuite and ELI5, hope you did too. Source code can be found at [Github](#). Have a great week!

References:

[sklearn-crfsuite](#)[ELI5](#)

Machine Learning

NLP

Natural language processing

Python

Named Entity Recognition

More from the list: "NLP"

Curated by Himanshu Birla



Jon Gi... in Towards Data ...

Characteristics of Word Embeddings



· 11 min read · Sep 4, 2021



Jon Gi... in Towards Data ...

The Word2vec Hyperparameters

· 6 min read · Sep 3, 2021



Jon Gi... in

The Word2ve



· 15 min rea

[View list](#)



Written by Susan Li

Following



28K Followers · Writer for Towards Data Science

Changing the world, one post at a time. Sr Data Scientist, Toronto Canada.

<https://www.linkedin.com/in/susanli/>

More from Susan Li and Towards Data Science





Susan Li in Towards Data Science



Antonis Makropoulos in Towards Data Science

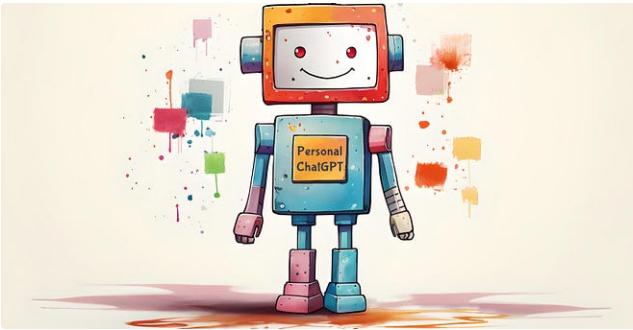
An End-to-End Project on Time Series Analysis and Forecasting...

Time series analysis comprises methods for analyzing time series data in order to extract...

9 min read · Jul 9, 2018

9.3K

94



Robert A. Gonsalves in Towards Data Science

Your Own Personal ChatGPT

How you can fine-tune OpenAI's GPT-3.5 Turbo model to perform new tasks using you...

· 15 min read · Sep 8

595

7



Susan Li in Towards Data Science

Multi-Class Text Classification Model Comparison and Selection

Natural Language Processing, word2vec, Support Vector Machine, bag-of-words, dee...

7 min read · Sep 25, 2018

3.5K

37

[See all from Susan Li](#)
[See all from Towards Data Science](#)

Recommended from Medium



 Angelica Lo Duca  in Towards Data Science

How to Create a Custom NER in Spacy 3.5

A quick tutorial on extracting custom entities from a text

◆ · 5 min read · Apr 25

 42  2

 Devang Chav... in Artificial Intelligence in Plain En...

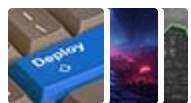
Custom Named Entity Recognition with Spacy

Spacy offers a robust open-source pipeline for natural language processing. So here I wi...

6 min read · Apr 6

Lists



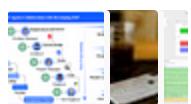
Predictive Modeling w/ Python

20 stories · 452 saves



Practical Guides to Machine Learning

10 stories · 519 saves



Natural Language Processing

669 stories · 283 saves



Coding & Development

11 stories · 200 saves

Artificial Intelligence (AI ORG), an ever-evolving field, has witnessed remarkable growth since its inception. Dating back to the Dartmouth Conference ORG in 1956 DATE, AI ORG has emerged as a multidisciplinary domain encompassing machine learning, natural language processing (NLP ORG), computer vision, and robotics. Recent breakthroughs, like the introduction of deep learning techniques in the early 2010s DATE, have accelerated AI ORG advancements. Tech giants like Google ORG, IBM ORG, and Microsoft ORG have invested heavily in AI ORG research and development. Significant milestones include the landmark victory of IBM ORG's Deep Blue ORG over Garry Kasparov PERSON in 1997 DATE and the emergence of voice assistants like Apple ORG's Siri in 2011 DATE. AI ORG continues to shape industries across healthcare, finance, and transportation, fueling innovation and transforming the way we live and work.



 Sahithi Reddy

Named Entity Recognition using Python and Spacy.

NER, short for Named Entity Recognition, is an NLP method that detects and categorizes...

4 min read · Jun 11

 24 

TIME 10:46 - PERIOD 2nd) (FORMATION Shotgun) (PLAYER J.Hurts (EVENT pass incomplete
DIRECTION short left to PLAYER A.Brown PENALTY on TEAM TEN) - (PLAYER B.Dupree Roughing the
asser, QUANTITY 15 yards, enforced at TEAM TEN QUANTITY 49) - No Play.

TIME 1:49 - PERIOD 2nd) (FORMATION Punt formation) Direct snap to (PLAYER C.Moore).
PLAYER C.Moore DIRECTION left end to TEAM DET QUANTITY 46 for QUANTITY 13 yards
PLAYER K.Crossen; (PLAYER E.Campbell).

TIME 1:06 - PERIOD 3rd) (FORMATION Shotgun) PENALTY on TEAM TB - (PLAYER R.Nunez-Roches
Neutral Zone Infraction, QUANTITY 5 yards, enforced at TEAM TB QUANTITY 21) - No Play.

 dp

Building Custom Named-Entity Recognition (NER) Models

Complete walk-through where we start with a dataset, quickly annotate our dataset...

6 min read · Apr 29

 25 

 Alex Reed

Writing Your First NLP Python Script: From Text Preprocessing t...

Welcome to the world of Natural Language Processing (NLP)! NLP is a fascinating field a...

5 min read · Sep 22

 56 

I.p
void 0,[tim...
r.writable(r.com...
,e){n.call(r,t,e)).then(e...
,e.rejection;e?console.error('Un...
,t,rejection;e instanceof Error?e.stack:void 0):c...
ue:",e,e instanceof Error?e.stack:{throw t}):return
;try{t.zone.runGuarded(function(){var t=1;return
=null,k=0,w=1,S=0,T=function(){var D=h("...
="object",P="function",M=h("currentTask"),D=h("...
new Error("Must be an instanceof Promise."),t.resolve
ZoneAwarePromise() { [native code] }},t.resolve
ation(t){function e(t){a&&(a=r(t)){var s=c.value;o...
done;c=u.next()){var s=c.value;o(s);l...
throw f.error}}return a;var s=c.value;o(s);l...
return l.ca...
his[y]

 Sanjithkumar

Text Preprocessing Part—2

Text preprocessing is an integral part of Natural Language Processing as no machine...

5 min read · Sep 7

 27 

See more recommendations