Piidetect_Piicatcher_piianalyzer

November 20, 2021

0.1 PIIDETECT

https://github.com/edwardcooper/piidetect

[ ]:
```
#if some packages give trouble then downgrade those packages to what was␣
 ↪available in 2019
import pandas as pd
```

[ ]:
```
#from piidetect.fakepii import Fake_PII
#fake_ = Fake_PII()
#fake_.create_fake_profile(10)
#train_labels, train_text, train_PII = fake_.create_pii_text_train(n_text =␣
 ↪5)#creating fake dataset
```

```
Finished creating fake profiles.
```

```
100%|      | 35/35 [00:00<00:00, 18304.32it/s]
```

[ ]:
```
import joblib
data_train = pd.read_csv("train.csv")
data_test = pd.read_csv("test.csv")

from piidetect.pipeline import word_embedding
model = word_embedding(algo_name = "word2vec",size = 100, min_count = 1,␣
 ↪workers =2)
model.fit(data_train['Text'])
```

```
Building new vocabulary and training the word2vec model
```

[ ]:
```
word_embedding(workers=2)
```

[ ]:
```
import joblib
data_train = pd.read_csv("train.csv")
data_test = pd.read_csv("test.csv")

from piidetect.pipeline import word_embedding
model = word_embedding(algo_name = "word2vec",size = 100, min_count = 1,␣
 ↪workers =2)
model.fit(data_train['Text'])
```

[ ]:
```
data_test
```

[ ]:
```
                                                  Text   Labels  \
0    Property long both group. Pass office Apt. 100…  Address
1    Occur attorney summer after heavy. Professor P…  Address
2    Apt. 100 Challenge investment forget continue …  Address
3    Continue across recognize exist fish. You Apt…  Address
4    Newspaper long everybody police. Service deter…  Address
..                                                 …      …
155  State class memory kid sister to each. Poor fo…     None
156                  Think tend herself ok reveal as.     None
157  Material total home offer who stage paper. Pla…     None
158  Article drug protect he free price same. Dinne…     None
159  Whose middle contain back ground top. Standard…     None

                    PII
0             Apt. 100
1    48756 Palmer Wells
2             Apt. 100
3             Apt. 704
4       7943 Daniel Row
..                  …
155              None
156              None
157              None
158              None
159              None

[160 rows x 3 columns]
```

[ ]:
```
#creating new label with 0 and 1
from piidetect.pipeline import binary_pii
data_train['Target'] = data_train['Labels'].apply(binary_pii)
```

[ ]:
```
data_train
```

[ ]:
```
                                                  Text   Labels  \
0    Option protect 0070 Steven Lodge away just mem…  Address
1    Large street quality subject figure such. 5103…  Address
2    84205 Jonathan Well Suite 322 West Kellyberg, …  Address
3    Small any occur level. Option third his set bl…  Address
4    Determine sit various quite as present. Watch …  Address
..                                                 …      …
235            Give until smile necessary fly street.     None
236  Floor arrive reality image listen. Sister cond…     None
237  Now open war center with. Television picture w…     None
238  Evening close live now on there. Area put onto…     None
```

```
239       Approach my party. Reality table home maybe.     None

                                                    PII  Target
0                                        0070 Steven Lodge       1
1    5103 Alyssa Junction Suite 026 South Matthew, …       1
2    84205 Jonathan Well Suite 322 West Kellyberg, …       1
3                                             Apt. 483       1
4                                            Suite 249       1
..                                                 …     …
235                                             None       0
236                                             None       0
237                                             None       0
238                                             None       0
239                                             None       0

[240 rows x 4 columns]
```

[ ]:
```
#using logistic regression
from piidetect.pipeline import word_embedding, text_clean
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression

logit_clf_word2vec = LogisticRegression(solver = "lbfgs", max_iter = 10000)

word2vec_pipe = Pipeline([('text_cleaning', text_clean()),
                 ("word_embedding", word_embedding(algo_name = "word2vec",␣
 ↪workers =2)),
                 ("logit_clf_word2vec",logit_clf_word2vec)
                ])

word2vec_pipe.fit(data_train["Text"],data_train['Target'])#fitting the model on␣
 ↪the training data
```

```
Building new vocabulary and training the word2vec model
transforming while training word2vec model with new data.

  0%|          | 0/240 [00:00<?,
?it/s]/Users/priankaball/Desktop/myenv/lib/python3.8/site-
packages/gensim/models/keyedvectors.py:877: FutureWarning: arrays to stack must
be passed as a "sequence" type such as list or tuple. Support for non-sequence
iterables such as generators is deprecated as of NumPy 1.16 and will raise an
error in the future.
  vectors = vstack(self.word_vec(word, use_norm=True) for word in
used_words).astype(REAL)
100%|      | 240/240 [00:00<00:00, 12276.01it/s]
100%|      | 240/240 [00:00<00:00, 170557.94it/s]
```

[ ]:
```
Pipeline(steps=[('text_cleaning',
                 <piidetect.pipeline.text_clean object at 0x7fe04dd4b6a0>),
                ('word_embedding', word_embedding(workers=2)),
                ('logit_clf_word2vec', LogisticRegression(max_iter=10000))])
```

[ ]:
```
#hyper parameter tuning
#this parts takes too long so we can skip
from sklearn.model_selection import RandomizedSearchCV
from piidetect.pipeline import word_embedding, text_clean
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
import random

#logit_clf_word2vec = LogisticRegression(solver = "lbfgs", max_iter = 10000)

#pipe = Pipeline([('text_cleaning', text_clean()),
#                 ("word_embedding", word_embedding( workers =2)),
#                 ("logit_clf_word2vec",logit_clf_word2vec)
#                ])


#param_grid = {
#    'word_embedding__algo_name':['word2vec', 'doc2vec','fasttext'],
#    'word_embedding__size':[100,200,300],
#    'logit_clf_word2vec__C': range(0,10),
#    'logit_clf_word2vec__class_weight':[{0: 0.9, 1: 0.1}, {0: 0.8, 1: 0.2}, {0:
 ↪ 0.7, 1: 0.3},None]
#}

#pipe_cv = RandomizedSearchCV(estimator = pipe,param_distributions =␣
 ↪param_grid,\
#                                    cv =10, error_score = 0,n_iter = 10 ,␣
 ↪scoring = 'f1'\
#                                    ,return_train_score=True, n_jobs = 1)

#pipe_cv.fit(data_train["Text"],data_train['Target'])
#print(pipe_cv.best_estimator_)
```

[ ]:
```
#dumping pipeline after training
#import joblib
#joblib.dump(pipe_cv.best_estimator_, 'pipe_cv.pkl', compress = 1)
```

0.2 PIICatcher

https://github.com/tokern/piicatcher

[ ]:
```
import piicatcher
from piicatcher import scan_database

dir(piicatcher)
```

[ ]:
```
['__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__path__',
 '__spec__',
 '__version__',
 'api',
 'catalog',
 'explorer',
 'log_mixin',
 'piitypes',
 'scan_database',
 'scanner']
```

[ ]:
```
#import scanner from piicatcher

from piicatcher import scanner
from piicatcher import piitypes

"""Different types of scanners for PII data"""
import logging
import re
from abc import ABC, abstractmethod

import spacy
from commonregex import CommonRegex

from piicatcher.piitypes import PiiTypes


# pylint: disable=too-few-public-methods
class Scanner(ABC):
#"""Scanner abstract class that defines required methods"""

    @abstractmethod
    def scan(self, text):
#        """Scan the text and return an array of PiiTypes that are found"""
        pass
```

```python
class RegexScanner(Scanner):
#  """A scanner that uses common regular expressions to find PII"""

    def scan(self, text):
        """Scan the text and return an array of PiiTypes that are found"""
        regex_result = CommonRegex(text)

        types = []
        if regex_result.phones:  # pylint: disable=no-member
            types.append(PiiTypes.PHONE)
        if regex_result.emails:  # pylint: disable=no-member
            types.append(PiiTypes.EMAIL)
        if regex_result.credit_cards:  # pylint: disable=no-member
            types.append(PiiTypes.CREDIT_CARD)
        if regex_result.street_addresses:  # pylint: disable=no-member
            types.append(PiiTypes.ADDRESS)

        #return types
        return print("RegexScan:", text,list(types))

class NERScanner(Scanner):
    """A scanner that uses Spacy NER for entity recognition"""

    def __init__(self):
        self.nlp = spacy.load("en_core_web_sm")

    def scan(self, text):
        """Scan the text and return an array of PiiTypes that are found"""
        logging.debug("Processing '%s'", text)
        doc = self.nlp(text)
        types = set()
        for ent in doc.ents:
            logging.debug("Found %s", ent.label_)
            if ent.label_ == "PERSON":
                types.add(PiiTypes.PERSON)

            if ent.label_ == "GPE":
                types.add(PiiTypes.LOCATION)

            if ent.label_ == "DATE":
                types.add(PiiTypes.BIRTH_DATE)

        logging.debug("PiiTypes are %s", ",".join(str(x) for x in list(types)))
        #return list(types)
        return print("NERScanner:", text,list(types))
```

```
class ColumnNameScanner(Scanner):
    regex = {
        PiiTypes.PERSON: re.compile(
            "^.*(firstname|fname|lastname|lname|"
            "fullname|maidenname|_name|"
            "nickname|name_suffix|name).*$",
            re.IGNORECASE,
        ),
        PiiTypes.EMAIL: re.compile("^.*(email|e-mail|mail).*$", re.IGNORECASE),
        PiiTypes.BIRTH_DATE: re.compile(
            "^.*(date_of_birth|dateofbirth|dob|"
            "birthday|date_of_death|dateofdeath).*$",
            re.IGNORECASE,
        ),
        PiiTypes.GENDER: re.compile("^.*(gender).*$", re.IGNORECASE),
        PiiTypes.NATIONALITY: re.compile("^.*(nationality).*$", re.IGNORECASE),
        PiiTypes.ADDRESS: re.compile(
            "^.*(address|city|state|county|country|"␣
 ↪"zipcode|postal|zone|borough).*$",
            re.IGNORECASE,
        ),
        PiiTypes.USER_NAME: re.compile("^.*user(id|name|).*$", re.IGNORECASE),
        PiiTypes.PASSWORD: re.compile("^.*pass.*$", re.IGNORECASE),
        PiiTypes.SSN: re.compile("^.*(ssn|social).*$", re.IGNORECASE),
    }

    def scan(self, text):
        types = set()
        for pii_type in self.regex:
            if self.regex[pii_type].match(text) is not None:
                types.add(pii_type)

        return print("ColumnNameScanner:", text,list(types))

#print(RegexScanner().scan("610-504-0413"))
#NERScanner().scan("Tommy is out. Jason is in Great Britain")
```

[ ]:
```
with open('PII_df.txt') as f:
  print(piicatcher.scan_file_object(f))
```

[ ]:
```
scan_type = [ColumnNameScanner(), NERScanner(), RegexScanner()]

for Scanner in scan_type:
    Scanner.scan(text = 'My phone number is 610-504-0413')
```

```
ColumnNameScanner: My phone number is 610-504-0413 []
NERScanner: My phone number is 610-504-0413 []
RegexScan: My phone number is 610-504-0413 [<PiiTypes.PHONE: 3>]
```

0.3 PIIAnalyzer

https://gitlab.math.ubc.ca/tomyerex/piianalyzer

[ ]:
```
from nltk.tag import StanfordNERTagger


stanford_ner_dir = 'stanford-ner/' # download file from here https://nlp.
 ↪stanford.edu/software/CRF-NER.shtml#Download and then set directory where␣
 ↪the file is
eng_model_filename= stanford_ner_dir + 'classifiers/english.conll.4class.
 ↪distsim.crf.ser.gz'
my_path_to_jar= stanford_ner_dir + 'stanford-ner.jar'

st = StanfordNERTagger(model_filename=eng_model_filename,␣
 ↪path_to_jar=my_path_to_jar)
#st.tag('Rami Eid is studying at Stony Brook University in NY'.split()) #example
```

[ ]:
```
import csv
from commonregex import CommonRegex
from nltk.tag.stanford import StanfordNERTagger


class PiiAnalyzer(object):
    def __init__(self, filepath):
        self.filepath = filepath
        self.parser = CommonRegex()
        #self.standford_ner = StanfordNERTagger('classifiers/english.conll.
 ↪4class.distsim.crf.ser.gz')
        self.standford_ner = st

    def analysis(self):
        people = []
        organizations = []
        locations = []
        emails = []
        phone_numbers = []
        street_addresses = []
        credit_cards = []
        ips = []
        data = []
```

```
        with open(self.filepath, newline='') as filedata:
            reader = csv.reader(filedata)

            for row in reader:
                data.extend(row)
                for text in row:
                    emails.extend(self.parser.emails(text))
                    phone_numbers.extend(self.parser.phones("".join(text.
↪split())))
                    street_addresses.extend(self.parser.street_addresses(text))
                    credit_cards.extend(self.parser.credit_cards(text))
                    ips.extend(self.parser.ips(text))


        for title, tag in self.standford_ner.tag(set(data)):
            if tag == 'PERSON':
                people.append(title)
            if tag == 'LOCATION':
                locations.append(title)
            if tag == 'ORGANIZATION':
                organizations.append(title)

        return {'people': people, 'locations': locations, 'organizations':␣
↪organizations,
                'emails': emails, 'phone_numbers': phone_numbers,␣
↪'street_addresses': street_addresses,
                'credit_cards': credit_cards, 'ips': ips
                }
```

[ ]:
```
PiiAnalyzer('pii.csv').analysis()
```

[ ]:
```
{'people': ['Sam', 'Country', 'Roberts'],
 'locations': ['United', 'States', 'Benin'],
 'organizations': ['Michael', 'Email'],
 'emails': ['mroberts2@pbs.org',
  'awagner3@altervista.org',
  'mwagner4@zimbio.com'],
 'phone_numbers': ['0796477389', '9-(937)171-5306', '4-(374)794-1813'],
 'street_addresses': [],
 'credit_cards': [],
 'ips': ['72.141.150.39', '247.65.204.78', '202.9.208.160']}
```

[ ]: